View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by ScholarWorks@UMass Amherst

University of Massachusetts Amherst

ScholarWorks@UMass Amherst

Masters Theses

Dissertations and Theses

October 2019

Calculation of Scalar Isosurface Area and Applications

Kedar Prashant Shete

Follow this and additional works at: https://scholarworks.umass.edu/masters_theses_2

 Part of the Acoustics, Dynamics, and Controls Commons, Computer-Aided Engineering and Design Commons, and the Heat Transfer, Combustion Commons

Recommended Citation

Shete, Kedar Prashant, "Calculation of Scalar Isosurface Area and Applications" (2019). *Masters Theses*. 854.
https://scholarworks.umass.edu/masters_theses_2/854

This Open Access Thesis is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Masters Theses by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

CALCULATION OF SCALAR ISO-SURFACE AREA AND APPLICATIONS

A Thesis Presented

by

KEDAR PRASHANT SHETE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE

September 2019

Mechanical and Industrial Engineering

© Copyright by Kedar Prashant Shete 2019

All Rights Reserved

CALCULATION OF SCALAR ISO-SURFACE AREA AND APPLICATIONS

A Thesis Presented

by

KEDAR PRASHANT SHETE

Approved as to style and content by:

Stephen de Bruyn Kops, Chair

David Schmidt, Member

Hari Balasubramanian, Member

Sundar Krishnamurty, Department Head
Mechanical and Industrial Engineering

DEDICATION

To all those who have had the time and patience to teach me

ACKNOWLEDGMENTS

I would like to thank Prof. Steve de Bruyn Kops for being a valuable mentor and for teaching me to think systematically about any task I undertake. His insight and clarity about science continues to inspire me. We thank Professors Duane Storti and Jim Riley for their valuable insights. High performance computing resources were provided through the U.S. Department of Defense High Performance Computing Modernization Program by the Army Engineer Research and Development Center and the Army Research Laboratory under Frontier Project FP-CFD-FY14-007.

I would like to thank Prof. Dragoljub (Beka) Kosanovic for providing me with a wealth of industry knowledge and experience. His patient mentorship and support is extremely valuable to me.

My sincere thanks to Prof. David Schmidt for his helpful comments and providing me resources to improve the quality and content of my thesis. I sincerely thank Prof. Hari Balasubramanian for spending time with me and providing me helpful suggestions to improve the clarity and accessibility of my thesis.

I am immensely grateful towards my family for all the love and support that they have always provided me. I thank my father Prashant for being a steadfast source of inspiration and energy. I thank my mother Jayanti for her unconditional love and for her helpful suggestions regarding my thesis. I thank my sister Aditi for her patience, understanding and love. I thank my mentor Janardan for teaching me and supporting me in my endeavors. I thank Emily Shankle for being a boundless source of energy and patience and for helping me improve my thesis. I thank Sherlock, my pet cavy, for looking at my thesis and clicking at the undesirable parts.

I also thank my friends and lab mates at UMass - Gavin, Felipe, Abhishek, Akhil, Ritvij, Ben, Harshad, Sreenivasa, Serena, Sachin and Ayush for supporting me and making my time at the university colorful and interesting.

ABSTRACT

CALCULATION OF SCALAR ISO-SURFACE AREA AND APPLICATIONS

SEPTEMBER 2019

KEDAR PRASHANT SHETE

B.E., BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCES - PILANI

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Stephen de Bruyn Kops

The problem of calculating iso-surface statistics in turbulent flows is interesting for a number of reasons, some of them being combustion modeling, entrainment through turbulent/non-turbulent interfaces, calculating mass flux through iso-scalar surfaces and the mapping of scalar fields. A fundamental effect of fluid turbulence is to wrinkle scalar iso-surfaces. A review of the literature shows that iso-surface calculations have primarily been done with geometric methods, which have challenges when used to calculate surfaces that have high complexity, such as in turbulent flows. In this thesis, we propose an alternative integral method and test it against analytical solutions. We present a parallelized algorithm and code to enable in-simulation calculation of iso-surface area. We then use this code to calculate area statistics for data obtained from Direct Numerical Simulations and make predictions about the variation of the iso-scalar surface area with Taylor Péclet numbers between 9.8 and 4429 and Taylor Reynolds numbers between 98 and 633.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **v**

ABSTRACT . . . . . **vii**

LIST OF TABLES . . . . . **x**

LIST OF FIGURES . . . . . **xi**

CHAPTER

1. INTRODUCTION **1**

2. BACKGROUND **5**

3. THEORETICAL BASIS FOR INTEGRAL METHODS **9**

4. EVALUATING THE AREA RATIO **11**

4.1 Marching Cubes 11
4.2 Integral Method 12

5. ALGORITHM, EQUATIONS, CODE AND TESTING **19**

5.1 Algorithm and Equations 19
5.2 Test Cases 22

5.2.1 Case 1: $\phi(x, y, z) = cos(x) + cos(y) + cos(z)$ 22
5.2.2 Case 2: $\phi(x, y, z) = cos(x) + y$ 23
5.2.3 Case 3: $\phi(x, y, z) = cos(x)$ 24
5.2.4 Case 4: $\phi(x, y, z) = x^2 + y^2 + 2z$ 25
5.2.5 Case 5: Potential Modifications to the Integral Method 28

6. DIRECT NUMERICAL SIMULATIONS **32**

7. ADDITIONAL NUMERICAL CONSIDERATIONS **36**

8. ISOSURFACE AREA FOR DNS OF ISOTROPIC HOMOGENEOUS TURBULENCE 41

9. CONCLUSIONS 44

APPENDICES

A. WHY WE NEED BINS LIMITS FROM x TO $x + \delta x$ 45
B. ERROR ANALYSIS FOR AREA CALCULATION WITH BINNING 46
C. RELATION BETWEEN AREA INTEGRAL AND VOLUME INTEGRAL EQUATION FOR ISO-SURFACE AREA 48
D. CALCULATING ERROR TERMS FOR AREA RATIO EQUATION 52
E. SCRIPTS USED FOR CHAPTERS 4 AND 5 54
F. ISO-SURFACE CALCULATION CLASS - KEDAR.C 74
G. SPECTRAL CODE DOCUMENTATION 106
H. CFD HUMOUR 107

BIBLIOGRAPHY 108

LIST OF TABLES

Table **Page**

6.1 Simulation parameters and statistics for the highest resolution of each case. N and $\kappa_{max}L_k$ are shown for each velocity field at it highest resolution ($Sc = 7$); lower resolution is used for the cases with lower Sc in simulations R3, R4 and R5. 34

LIST OF FIGURES

Figure **Page**

4.1 Iso-surface for iso-value $\psi = 2$ for the test case $\phi(x, y, z) = cos(x) + cos(y) + cos(z)$ 12

4.2 Steps in a Marching Cubes Algorithm as given by Steps in a marching cube algorithm as given by Ben Anderson, Department of Computer Science, Carleton College 13

4.3 Relative error with marching cubes and integral method for iso-surface given by $\psi = 2$ and dimensionless bin width of $10^{-2}/6$. The area value obtained by marching cubes with a grid of 550^3 was used as the "exact" result to calculate error. 15

4.4 P.d.f convergence with fixed bin width and the Freedman-Diaconis rule. The plot shows that a bin width of $\Delta\phi = 0.01$ is sufficiently small to generate a good estimate for the p.d.f. that agrees with the estimate made by the Freedman-Diaconis rule, which uses reducing bin sizes. 15

4.5 Points in bin with fixed bin width and the Freedman-Diaconis rule. 16

4.6 Changing fidelity on iso-surface $\psi = 2$ with fixed grid size N and varying bin width $\Delta\phi$. The plot shows that with a given size of data, there is an optimum bin width that provides the best accuracy. Larger data sets allows for a smaller bin width, since we have more points on the layer $V(\psi, \Delta\phi)$ 16

4.7 Convergence of p.d.f $P(\phi; \phi = \psi)$ for $\psi = 2$ as number of samples 17

4.8 Convergence of gradient $\langle|\nabla\phi|\rangle_{V(\psi,\Delta\phi)}$ with increasing grid size. It can be seen that the gradient converges faster than the p.d.f, by comparing this to 4.4 18

5.1 Initial and Phase Shifted Scalar Data Set $\boldsymbol{\phi}$ 20

5.2 Algorithm for Monte Carlo Method 21

5.3 Convergence rates for various sampling methods (Numerical Recipes, The Art of Scientific Computing) 22

5.4 Convergence of area ratio plotted with number of points on the layer, with three grid sizes for scalar value $\psi = 2$ and a bin width $\Delta\phi = 0.001$. It can be seen that all three calculations converge to the same result, which means that accuracy does not depend upon starting grid size. 23

5.5 Convergence of area ratio plotted with number of points on the layer, for scalar value $\psi = 0$ with three bin widths 0.01, 0.001 and 0.0001. It can be seen that the area ratio converges for arbitrary bin width, given sufficient number of samples 24

5.6 Convergence of area ratio plotted with number of points on the layer, with three bin widths 0.1, 0.01 and 0.001. The area ratio converges to the analytical result. 25

5.7 Convergence of $A(\psi)$ with 800,000 interpolations 26

5.8 Convergence of $A(\psi)$ with 800,000 interpolations, $-2/3$ slope 26

5.9 Relative error in the estime for the area of the iso-surface in the simple example with $\psi = 4$. 27

5.10 First order convergence of area ratio with bin width using left sided bins (panel (a)) and Second order convergence of area ratio with bin width using centered bins (panel (b)). This agrees with the expected convergence for the rectangle and midpoint rules for numerical integration. 30

5.11 Optimum bin width determined by testing for different number of samples. The line represents the optimum bin width as predicted by (5.2) adjusted to the rightmost point. It can be seen that the calculations agree broadly with the prediction, but there is a certain amount of scatter. 31

7.1 Error in the area of one iso-surface ψ in case R3, $Pr = 7$, for various nondimensional bin widths $\widetilde{\Delta\phi}$ (panel (a)) and grid resolutions (panel (b)). The dotted line is a reference slope. 39

8.1 The iso-surface areas normalized by the area of a horizontal plane in the simulations. The symbols mark the average for 2 iso-values and the bars indicate the range from the largest to smallest areas. The dotted line is a reference line, not a fitted line. 43

C.1 Cartoon of an iso-surface in a volume to illustrate the notation. 49

D.1 Area for the test function in §3 was calculated with a bin width 0.01, i.e. between $\psi = 4$ and $\psi = 4.01$. The plot shows the area calculated with the exact ((C.10)) and simplified ((C.11)) equations given in §3 53

CHAPTER 1

INTRODUCTION

The area of an iso-surface in a turbulent flow reflects the instantaneous balance between advection that stretches and compresses the iso-surface, diascalar diffusion that smooths the iso-surface, and possibly the effects of other phenomena such as chemical reaction. This is true whether the scalar is passive, active or inherent to the velocity field, e.g., enstrophy. Thus the iso-surface area is a composite measure that is of fundamental interest because of what it can tell us about fluid turbulence. There are also specific applications requiring knowledge of an iso-surface area including the flame area in combustion [8, 68], the size of the entrainment area in a turbulent jet [15, 57], and the area of the turbulent/non-turbulent interface [10, 70, 73]. As noted by Kim and Bilger [30], computing the iso-surface is a simplified case of the more general problem of finding the flux of one scalar through the iso-surface of another, which is inherent in all flow configurations involving entrainment.

A principal motivation for understanding iso-surfaces is combustion modeling. In both premixed and non-premixed combustion, the rate of chemical reaction depends on the volume in which the species concentrations and temperature are suitable for reaction. It also typically depends on the mass flux of one scalar through the iso-surface of another. Calculating either of these quantities is a generalization of the problem of finding the area of one or more iso-surfaces [30, 47]. It is, however, impractical to resolve iso-surfaces in simulations that are practical for the design of combustion devices, and so these quantities must be parameterized. One avenue of parametrization is via the iso-surface area to volume ratio as a function of flow parameters includ-

ing the Reynolds, Schmidt, and Damköholer numbers, etc. Peters [46] explored the importance of calculating iso-scalar surfaces and their role in combustion modeling. One of the earlier attempts of modeling combustion using iso-scalar surface statistics is given in Bray and Swaminathan [6]. Swaminathan and Bray [69, pp. 41-60] discussed the equation for iso-surface area and its uses in modeling premixed combustion. Bray [5] studied laminar flamlets in premixed combustion. A more recent example of combustion analysis based on the iso-surface area is that of Chaudhuri et al. [8] who studied the geometric flame thickness of a reacting jet. Motivated by the modeling of premixed combustion, Dopazo et al. [19] studied the propagation of scalar iso-surfaces and also provides an equation for the time rate of change of the infinitesimal iso-surface area [19, equation 25]. Zheng et al. [79] studied the propagation of scalar iso-surfaces in isotropic turbulence and formulated a model for estimating iso-surface area changes with time in order to model a planar premixed flame front. Turbulent combustion models have proven to be of use for the design of combustion devices ([48]).

Applications for iso-surface area calculations outside the field combustion include the size of the entrainment area in a turbulent jet [15, 57] and the area of the turbulent/non-turbulent interface [10, 70, 73]. Schumacher et al. [63] considered the relationships between the geometrical and statistical properties of a scalar field. In general, these quantities are important for understanding and modelling mixing environmental and technological flows. Watanabe et al. [73] consider internal wave energy transported through an isosurface of enstrophy in a stably stratified wake. The mixing of passive scalars in the presence of turbulent motion is a subject of great theoretical and practical interest and has applications in reacting flows and combustion, mixing of salt and plankton in oceans and of pollutants in the atmosphere, as well as mixing of biological substances. Much experimental effort has been spent in mapping scalar fields in three dimensions [61].

The problem of finding mass flux through level scalar surfaces was studied by [29, equation 2.9]. It can be observed that the equation for the mass flux is very similar to the equation for the iso-surface area in that both require calculation of a probability density function and a scalar gradient. In fact, the flux equation is derived using the area ratio equation and is identical to it if a unit density and velocity is used for its calculation (a justification for the area ratio calculation/formula has been provided in Chapter 3). In other words, the mass flux through a scalar iso-surface, with unit mass density and unit velocity, is the iso-surface area. This lends further importance to the development of accurate methods for calculating the area of level scalar sets.

In the foregoing, it is noted that advection both stretches and compresses the iso-surface with the net effect being that it increases the iso-surface area. In the case of velocity iso-surfaces, this is consistent with the Richardson cascade [56] and the resulting downscale transfer of energy to smaller length scales. Molecular diffusion acts at the small scales and prevents the iso-surface area from being unbounded. The hypothesis of Kolmogorov and Oboukhov [32, 37, 38] implies that, given sufficient separation between the energy-containing and dissipative length scales, that the iso-surface per unit volume will be universal for a velocity iso-surface, i.e., that there is an asymptotic state at sufficiently high Reynolds number. Research in IHT at higher Reynolds number is ongoing, but there is some evidence that when the Taylor Reynolds number Re_λ is about 150, the inertial range scaling of some quantities is observed and the flow exhibits characteristics of the high Reynolds number limit [1, 11, 76]. To our knowledge, there is no published result regarding whether velocity iso-surfaces are characteristic of high Reynolds number when $Re_\lambda \approx 150$, and we note that some characteristics of the velocity field change as the Reynolds number is increased above this value Ishihara et al. [27].

Corrsin [9], Oboukhov [39] introduced the hypothesis that, given sufficient scale separation, a scalar field, as well as the velocity field will exhibit universal statistical

characteristics. Sreenivasan [65], however, reports that "there is hardly any credible evidence that this asymptotic state is unique" for a scalar. He further notes that the asymptotic state is approached very slowly with increasing Reynolds number. Yeung et al. [76] report simulations of scalar mixing in IHT up to with scalar statistics related to surface area, e.g., dissipation rate of scalar variance, differing between $Re_\lambda \approx 240$ and $Re_\lambda \approx 700$ for Prandtl number $Pr = 1$. In very new DNS data of statistically stationary stably stratified shear flows, the scalar field approaches the asymptotic limit much more slowly than the velocity field as the Reynolds number increases [51]. In this thesis, the iso-surface area of a passive scalar both from the perspective of how to compute it accurately from direct numerical simulation (DNS) data and of how it varies with Reynolds and Prandtl number in isotropic homogeneous turbulence (IHT) has been considered.

CHAPTER 2

BACKGROUND

We think that the problem of finding the areas of scalar level sets is an interesting and important problem in fluid mechanics. In this chapter, we explore the existing algorithms and methods of calculating iso-surface areas. We divide these methods into two categories, geometric methods and integral methods.

Geometric methods are based on reconstructing the iso-surface from available data. Since the scalar data is available at discrete points, values between grid points can either be approximated for non periodic cases, or can be interpolated exactly and sampled using a truncated Fourier for periodic DNS data. These points are then connected by polygons and the area of the iso-surface is the combined area of all polygons. Marching Cubes, Marching Tetrahedra, and Surface Nets are some of the popular geometric iso-surface approximation methods. A review of the geometrical methods can be found in Patera and Skala [44]. Paraview, an open source data visualization software, uses the marching cubes algorithm for constructing iso-surfaces [60, Chapter 6]. We observed that such applications are mostly made for data visualization. Geometric methods like marching cubes do not always converge to the true surface area, sometimes have topological ambiguities that can result in connectivity problems and can also have high computational complexity([34]). Much effort has been put into resolving the ambiguity in the connectivity of points for the marching cubes algorithm. Lewiner et al. [33] studied the possible connection ambiguities in Marching Cubes and their results show that there can be connectivity issues even with relatively simple convex iso-surface geometries. Newman and Yi [36] conducted

a review of the marching cubes algorithm noted that there are many variations within it and a number of 'spin-off' methods to fix different kinds of problems. Level scalar surfaces can be highly wrinkled and disjoint in turbulent reacting flows and a direct geometrical representation may not be feasible [50]. Thus, there is a need for methods that can calculate areas and fluxes without constructing the scalar iso-surface.

An additional consideration when choosing a method for calculating iso-surfaces are advances in computer architecture. As early as 1988, Payne et al. [45] noted that massively parallel computers had enabled the calculation of flow fields that were previously intractable in simulations. Since then, the importance of parallel algorithms have increased dramatically, particularly since the breakdown of Dennard scaling in approximately 2006 [3, 16] and the resulting requirement that, for an algorithm to run faster, it must exploit additional parallelism on a single computer processor and across multiple processors. Raase and Nordström [54], for example, reported the utility of many-core processors for fluids simulations. Kolla and Chen [31], though, report that geometric methods for finding iso-surface areas are difficult to parallelize. Monte-Carlo techniques, suitable for evaluating integral methods for computing iso-surface areas, are well-suited for parallelization [17].

Integral methods are based on (3.6) which reduces sources of error in the calculation procedure by making it an entirely numerical problem of obtaining converged values for the components of the calculation. In other words, the method outlined in Chapter 3 can be proved to converge to the correct area ratio with increasing amount of data and reducing bin width. The actual calculation procedure varies in the evaluation of quantities in 3.6, but the foundation is common. Some examples of the use of these methods are [29, 42, 63]. Yurtoglu et al. [78] conducted convergence tests for a probability density-based Monte Carlo method to calculate the volume of a torus. They also provide a proof for the formula for surface integral of a function over a level scalar set [78, Equation 10] which when discretized for an infinitesimal scalar

range, goes over to the formula in Chapter 3. The iso-surface statistics summed over the scalar domain has also been a point of interest ([58]) and the area ratio formula in Chapter 3 agrees with their result, after integration over the entire range of scalar values. It should however be noted that integral methods do not provide an exact location of the iso-surface.

Massively parallel computers have enabled the analyst to solve complicated flow fields (turbulent, chemically reacting) that were previously intractable [45]. Examples like Raase and Nordström [54] demonstrate that numerical techniques have benefited greatly in their implementation due to the development of fast and parallel computers. Monte Carlo methods are well suited for parallelization, and Dimov et al. [17] is one of the many examples of their parallel implementation. In this thesis, a probability density based method to calculate the area ratio and a Monte Carlo based implementation algorithm have been proposed. Large datasets are required to run convergence tests. Periodic DNS data can be phase shifted in Fourier space to obtain exact scalar values and scalar gradients between grid points, which solves the problem of having enough data. The problem of finding roots of a Fourier series scales as N^3 which makes it an impractical problem to solve [4]. Thus, there was a need to develop a method that can converge with increasing number of samples and still provide a reasonable approximation for the iso-surface area.

In this thesis, we have attempted to bridge the gap between needs for speed and accuracy evident through existing literature, and existing methods to calculate iso-surface areas. For validation of the method and code, a test Fourier series was numerically integrated and the proposed method was tested against that solution. The resulting parallel calculation procedure for the area to volume ratio has been presented. The procedure involves the use of periodic test data to perform a convergence study for the area ratio and a Monte Carlo integration with FFT based sampling to accelerate convergence. The algorithm has then been parallelized and

implemented in the C++ language using MPI. The code has been merged with a DNS spectral code to allow in-simulation calculation of area statistics.

CHAPTER 3

THEORETICAL BASIS FOR INTEGRAL METHODS

Let us first define some notation. Consider a scalar field $\phi(\vec{x})$ to be a function that maps $\vec{x}$ to a ϕ. The volume in which we want the surface to volume ratio is V, the set of all possible position vectors in V is $\mathbf{X}$, and the set of all values of ϕ in V is $\boldsymbol{\phi}$. The subset of $\boldsymbol{X}$ at which $\phi = \psi$ is $\boldsymbol{X}_\psi$. Assuming that an isosurface exists with isovalue ψ then its area is $A(\psi)$ and the differential area associated with each point in $\boldsymbol{X}_\psi$ is $\hat{n}dA$ where $\hat{n} = \nabla\phi/\left|\nabla\phi\right|$. The volume between isosurfaces with $\phi = \psi$ and $\phi = \psi + d\phi$ corresponds to the set of all the position vectors between $\boldsymbol{X}_\psi$ and $\boldsymbol{X}_{\psi+d\phi}$. Consider a position vector $\vec{x} = \vec{\xi}$ such that $\vec{\xi} \in \boldsymbol{X}_\psi$. For every $\vec{\xi}$, since the gradient $\nabla\phi$ is defined at every point, there exists a differential vector $d\vec{x} = \hat{n}dx$ such that

$$\vec{x} = \vec{\xi} + d\vec{x} \in \boldsymbol{X}_{\psi+d\phi} \ . \tag{3.1}$$

Hence, a volume element at $\vec{\xi}$ corresponding to the volume between ensembles $\boldsymbol{X}_\psi$ and $\boldsymbol{X}_{\psi+d\phi}$ is given by $\hat{n}dA \cdot \hat{n}dx$. The volume of this region is given by integrating over $\boldsymbol{X}_\psi$:

$$dV = \int_{\boldsymbol{X}_\psi} \hat{n}dA \cdot \hat{n}dx \ . \tag{3.2}$$

Now consider a finite thickness of an iso-surface layer where the infinitesimal width $d\phi$ is replaced by a finite width $\Delta\phi$. The volume dV given in (3.2) integrated over the region between $\boldsymbol{X}_\psi$ and $\boldsymbol{X}_{\psi+\Delta\phi}$ gives us the volume of the finite layer between $\boldsymbol{X}_\psi$ and $\boldsymbol{X}_{\psi+\Delta\phi}$, which is represented by $V(\psi, \Delta\phi)$. Now consider Federer's Coarea formula that allows integrals over surfaces to be converted to integrals over volumes

which is given by (3.3). This particular presentation of the Coarea formula has been taken from Scheidegger et al. [58, equation 3].

$$\int_{\psi}^{\psi+\Delta\phi} \int_{\boldsymbol{X}_{\psi}} q(\vec{x}) dA \; d\phi = \int_{V(\psi,\Delta\phi)} q(\vec{x}) |\nabla\phi| dV \tag{3.3}$$

where $q(\vec{x})$ is any scalar function defined over the domain of ϕ. Note that $V(\psi, \Delta\phi)$ can be any volume the equals or is a subset of the domain of ϕ. The term on the right hand side can be written as the ensemble average of $q(\vec{x})$ over $V(\psi, \Delta\phi)$ times the volume $V(\psi, \Delta\phi)$. Putting $q(\vec{x}) = 1$ in (3.3), the inner integral becomes the area of $\boldsymbol{X}_{\psi}$. Dividing both sides by V, (3.3) simplifies to (3.4), which is an exact equation as it is a direct consequence of Federer's Coarea formula.

$$\frac{1}{V} \int_{\psi}^{\psi+\Delta\phi} A(\phi) d\phi = \frac{\langle |\nabla\phi| \rangle_{V(\psi,\Delta\phi)} V(\psi, \Delta\phi)}{V} \tag{3.4}$$

Approximating the outer integral using the rectangle rule and dividing both sides by V, we get (3.5).

$$\frac{A(\psi)}{V} \Delta\phi \approxeq \frac{\langle |\nabla\phi| \rangle_{V(\psi,\Delta\phi)} V(\psi, \Delta\phi)}{V} \tag{3.5}$$

The volume $V(\psi, \Delta\phi)$ represents all possible position vectors $\vec{x}$ such that $\phi(\vec{x}) \in [\psi, \psi + \Delta\phi)$. Since V is the domain of ϕ, it represents all possible position vectors that produce the range of ϕ. Thus, (3.5) can be simplified to give (3.6) which is a first order approximation(with respect to $\Delta\phi$) for the area to volume ratio of the iso-surface given by $\boldsymbol{X}_{\psi}$ and converges to the exact area ratio $A(\psi)$ as $\Delta\phi$ is reduced.

$$\frac{A(\psi)}{V} \approxeq \langle |\nabla\phi| \rangle_{V(\psi,\Delta\phi)} \frac{V(\psi, \Delta\phi)}{V \; \Delta\phi} \tag{3.6}$$

This equation has been evaluated numerically in the following chapters.

CHAPTER 4

EVALUATING THE AREA RATIO

In this chapter, we have compared the geometric method of marching cubes as implemented in the commercial software MATLAB(2018a) to an integral method. We used three dimensional data sets for robust testing and good visualization.

Let $\boldsymbol{X}$ be a domain over which ϕ is periodic. The periodicity of ϕ is not a requirement for area ratio formula, but it enables us to generate more data by sampling between grid points. Let the domain given by $x \in [-\pi, \pi], y \in [-\pi, \pi], z \in [-\pi, \pi)$ be represented as $\boldsymbol{X} \in [-\pi, \pi]$. Consider a scalar field $\phi(x, y, z) = cos(x)+cos(y)+cos(z)$ defined over $\boldsymbol{X} \in [-\pi, \pi)$ and its range given by $\boldsymbol{\phi} = \phi(\boldsymbol{X}) \in [-3, 3]$. Let the number of points defined along each axis be N. The total size of the field is hence N^3. A sample iso-surface given by $\psi = 2$ has been shown in figure 4.1.

4.1 Marching Cubes

Marching Cubes with linear interpolation for the triangulated surface seems to be the most popular geometric method for iso-surface calculation. A series of diagrams showing the steps for this algorithm can be seen in Figure 4.2. Using a marching cubes algorithm code provided by MATLAB, the area ratio was calculated and is given in Figure 4.3. We observed that for $\psi = 2$, the method converged to an area to volume ratio of 0.114115819935933 with 550^3 grid points.

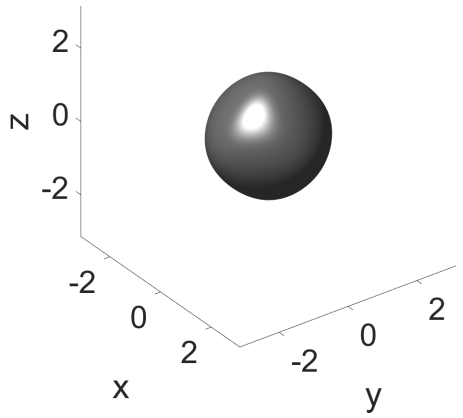

Figure 4.1: Iso-surface for iso-value $\psi = 2$ for the test case $\phi(x, y, z) = cos(x) + cos(y) + cos(z)$

4.2 Integral Method

In order to evaluate if obtaining a converged and accurate result is possible, we explored an area calculation method based on the integral expression for the area to volume ratio given by (3.6). The calculation of the area to volume ratio requires calculating two quantities, the quantity $\langle|\nabla\phi|\rangle_{V(\psi,\Delta\phi)}$ which is a volume averaged gradient magnitude and $P(\phi;\phi = \psi)$, which is the value of the p.d.f. of ϕ for the iso-value ψ.

To evaluate (3.6) for this test case, $\phi(\vec{x})$ is sampled at random locations chosen using a pseudo-random number generator with the term "pseudo-random" referring to how sequences are computed on a typical digital computer. There are advantages, though, to using a "quasi-random" sequence, such as that of Sobol' [64]; for a discussion see Press et al. [53, pp. 404-409]. As the samples are accumulated, the computation of $\langle|\nabla\phi|\rangle_{V(\psi,\Delta\phi)}$ is straightforward given $d\phi$ taking on a finite value $\Delta\phi$.

While the computation of $\langle|\nabla\phi|\rangle_{V(\psi,\Delta\phi)}$ is straightforward, the calculation of $P(\phi;\phi = \psi)$ requires some consideration. If we were to estimate the entire function $P(\phi)$ by sampling then we could construct a histogram by a variety of methods

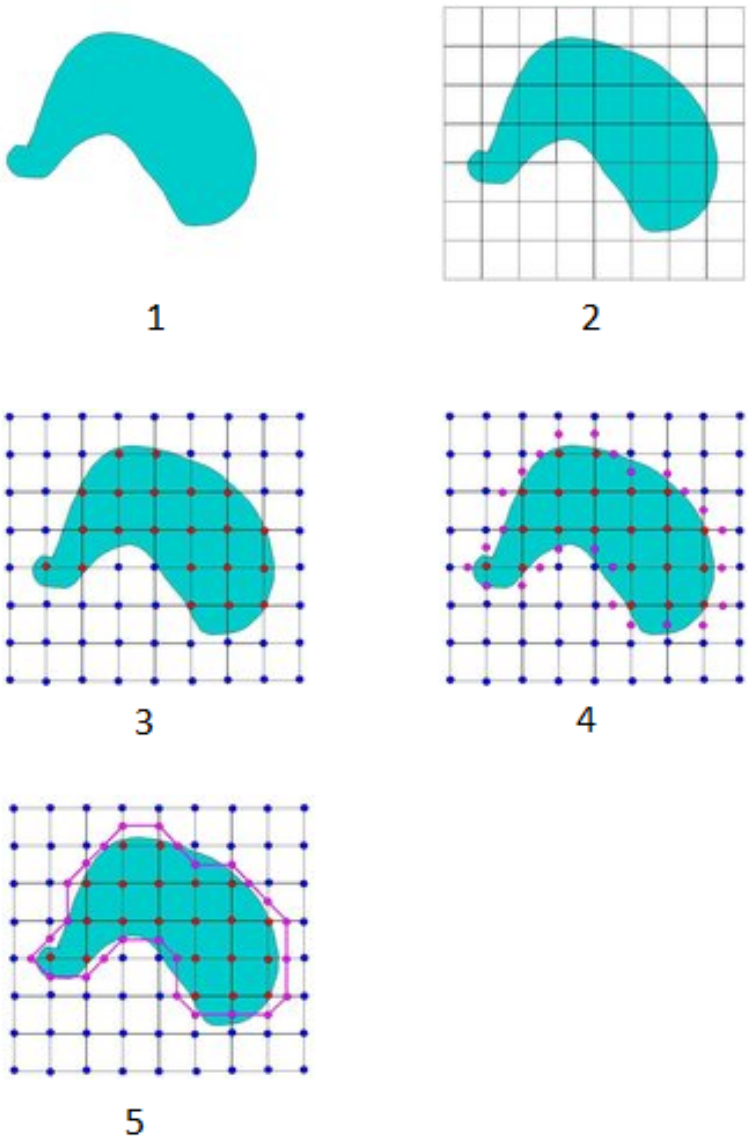

Figure 4.2: Steps in a Marching Cubes Algorithm as given by Steps in a marching cube algorithm as given by Ben Anderson, Department of Computer Science, Carleton College

[22]. For the current application, though, only $P(\phi;\phi=\psi)$ is needed so that a count of samples with $\psi \leq \phi(\vec{x}) < \psi + \Delta\phi$ is collected and then divided by the total number of samples and by $\Delta\phi$ to estimate $P(\phi;\phi=\psi)$. The probability density function calculation involves a selection of bin width $\Delta\phi$, which represents the fidelity of the captured iso-surface. This bin width also represents the spread of iso-values over which the average gradient $\langle|\nabla\phi|\rangle_{V(\psi,\Delta\phi)}$ is calculated. The probability density function calculated by selection of points in the bin width $[\phi, \phi+\delta\phi)$ is the probability density function for ϕ as the number of samples tends to infinity (A). Thus a smaller bin width should result in a higher accuracy of $P(\phi)$ given that there are enough sample points in the bin. Thus, the selection of bin width should be made based on the desired accuracy of the iso-surface and the sensitivity of the area to the bin width.

Relative Error for the area ratio calculated by the integral method for $\psi = 2$ and $\Delta\phi/(\phi_max - \phi_min) = 0.01/6$ has been shown in figure 4.3. It can be seen that with a bin width of $\Delta\phi = 0.01$, we get equal decimal places of precision, as expected from a method with first order error. A popular rule for calculating bin width is the Freedman-Diaconis Rule given by Freedman and Diaconis [22]. A comparison of PDF convergence calculated with fixed bin width and with this rule has been provided in Figure 4.4. The corresponding number of samples in the bin has been shown in Figure 4.5. The p.d.f's from the two methods converge to approximately the same value at 350^3 grid points, and the number of samples in the bin for a converged PDF are roughly 40,000, as can be seen from 4.5. From this, we conclude that fixing a fidelity for the iso-surface using $\Delta\phi$ assures a certain accuracy in the area ratio calculation, provided that the Monte Carlo integration is converged.

With a given amount of data, too small bin width sizes result in results that are not converged due to insufficient points in the bin, and too large bin width results in a result that is over averaged and far from the true value, as can be seen in Figure 4.6. The relative error in $P(\phi;\phi=2)$ as a function of number of points can be seen

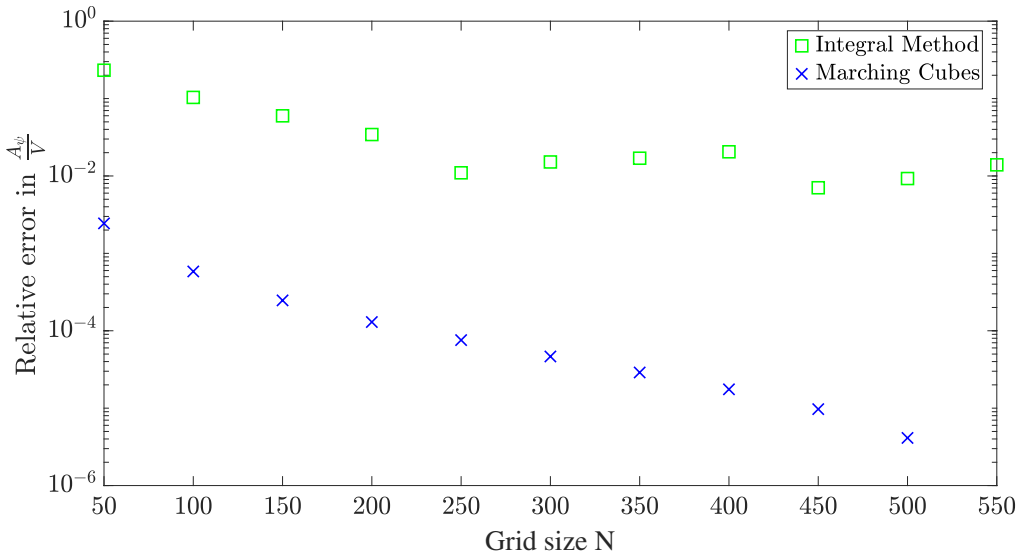

Figure 4.3: Relative error with marching cubes and integral method for iso-surface given by $\psi = 2$ and dimensionless bin width of $10^{-2}/6$. The area value obtained by marching cubes with a grid of 550^3 was used as the "exact" result to calculate error.

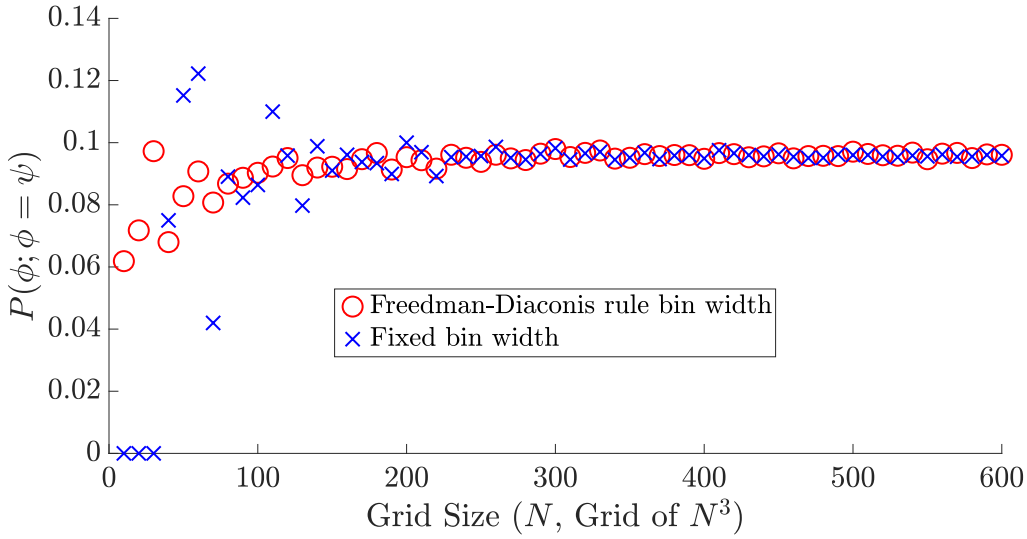

Figure 4.4: P.d.f convergence with fixed bin width and the Freedman-Diaconis rule. The plot shows that a bin width of $\Delta\phi = 0.01$ is sufficiently small to generate a good estimate for the p.d.f. that agrees with the estimate made by the Freedman-Diaconis rule, which uses reducing bin sizes.

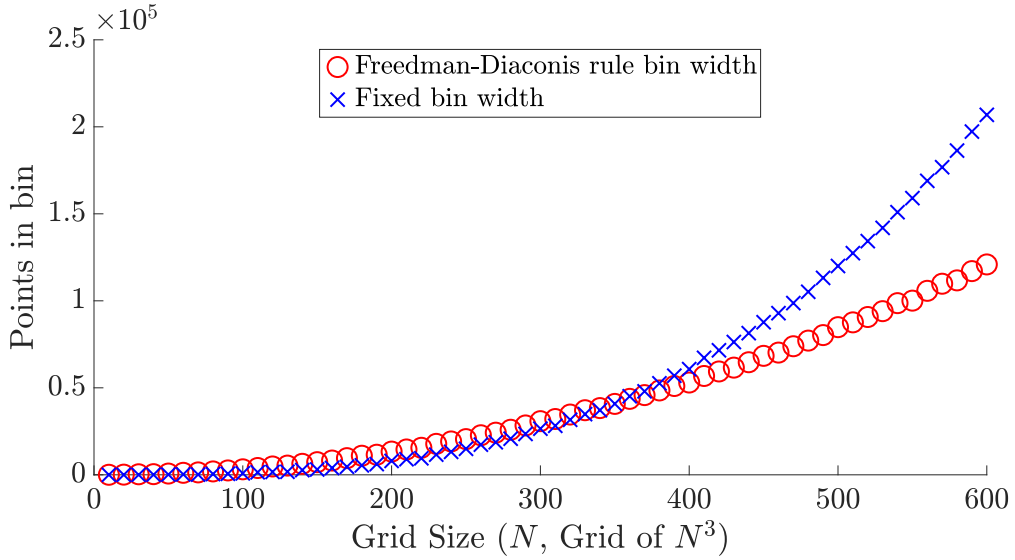

Figure 4.5: Points in bin with fixed bin width and the Freedman-Diaconis rule.

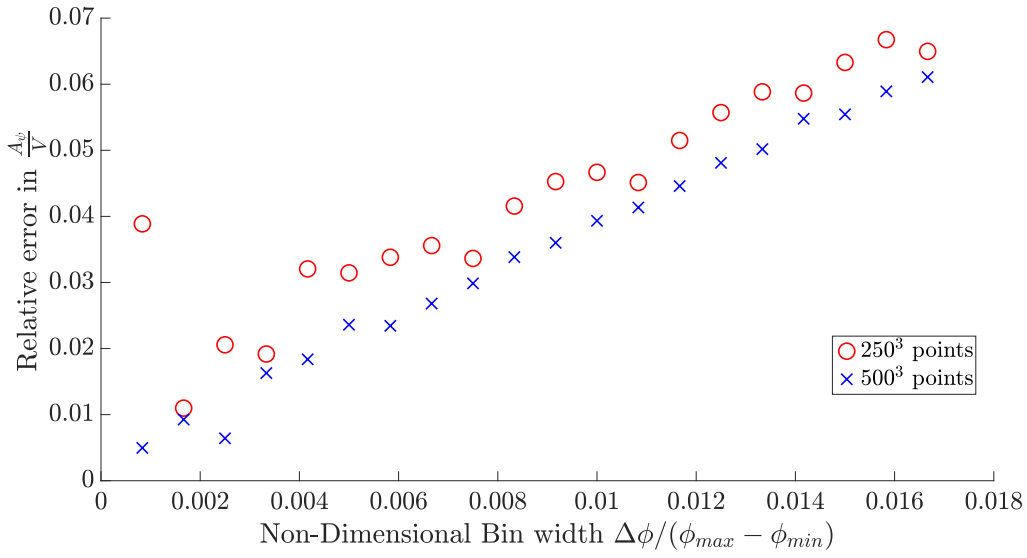

Figure 4.6: Changing fidelity on iso-surface $\psi = 2$ with fixed grid size N and varying bin width $\Delta\phi$. The plot shows that with a given size of data, there is an optimum bin width that provides the best accuracy. Larger data sets allows for a smaller bin width, since we have more points on the layer $V(\psi, \Delta\phi)$

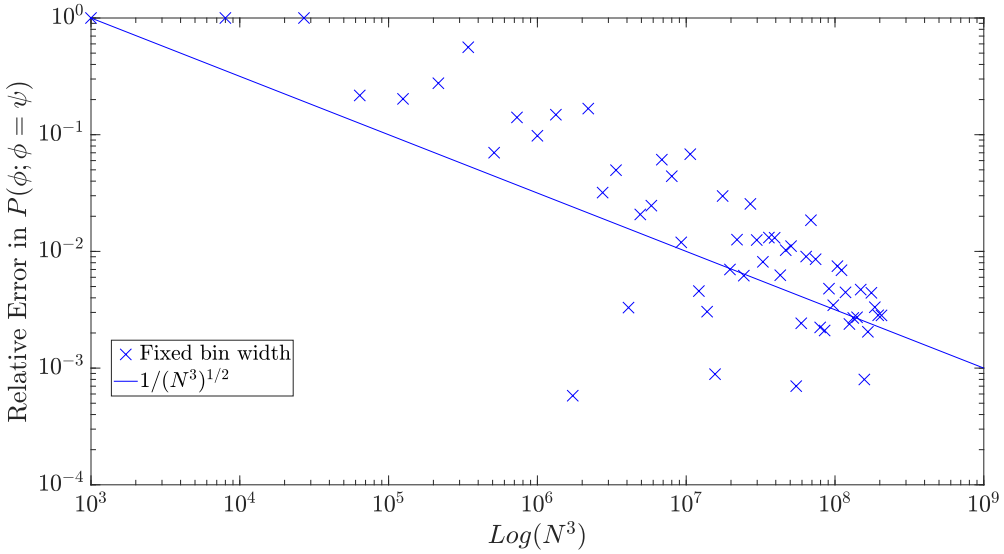

Figure 4.7: Convergence of p.d.f $P(\phi;\phi=\psi)$ for $\psi=2$ as number of samples

in Figure 4.7, which shows that the p.d.f converges as the square root of the number of samples as expected from theory [53].

The average gradient on the layer converges with increasing number of points as can be seen in Figure 4.8.

The scripts used to make these plots have been given in E. In the subsequent chapter, we provide the algorithm and equations for the paralleled C++ code that was developed in order to allow in-simulation calculation of iso-scalar surface statistics.

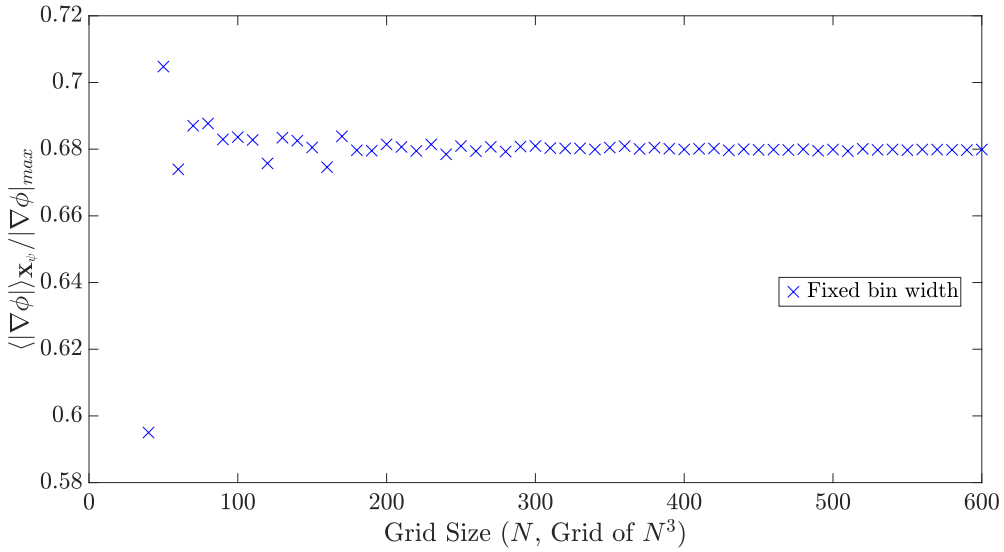

Figure 4.8: Convergence of gradient $\langle|\nabla\phi|\rangle_{V(\psi,\Delta\phi)}$ with increasing grid size. It can be seen that the gradient converges faster than the p.d.f, by comparing this to 4.4

CHAPTER 5

ALGORITHM, EQUATIONS, CODE AND TESTING

5.1 Algorithm and Equations

It can be seen from Figure 4.3 that the area ratio converges with an increase in number of points. Consider a data set of scalar $\phi(\boldsymbol{x})$ of size N^3 periodic over $\boldsymbol{x} \in [-\pi, \pi]$ which has sufficient grid points so that the field and gradients are fully resolved. In order to add more points to the area ratio calculation, $\phi(\boldsymbol{x})$ was phase shifted by a fraction of grid size and then differentiated to obtain the gradient field $\nabla\phi$. This is done using a three dimensional Discrete Fourier Transform of the scalar field making the phase shifted field and gradients exact. Figure 5.1 shows the initial and phase shifted grids. Note that the figure shows a phase shift in two dimensions for simplicity, but the actual phase shift was in three dimensions. The phase shift for each interpolation was specified using a random number generator to accelerate convergence and eliminate any bias that may occur due to bin positioning.

Let $f_x(\boldsymbol{\phi})$ be the Discrete Fourier Transform (DFT) of $\boldsymbol{\phi}$ in the x direction. Let $f_x(\phi(x,y,z))_i$ be i^{th} member of $f_x(\boldsymbol{\phi})$ that corresponds to the point (x,y,z) The subscript i implies that the point referred to is the i^{th} point in x direction. Thus, k_i is the wave number corresponding to that point in the same direction.
Let $r_i = Re(f_x(\phi(x,y,z))_i)$ and $j_i = Im(f_x(\phi(x,y,z))_i)$

$$f_x(\boldsymbol{\phi}) = FFTX(\boldsymbol{\phi}) \tag{5.1a}$$

$$Re(f_x(\phi(x+\Delta x, y, z))_i) = r_i cos(k_i \Delta x) - j_i sin(k_i \Delta x) \tag{5.1b}$$

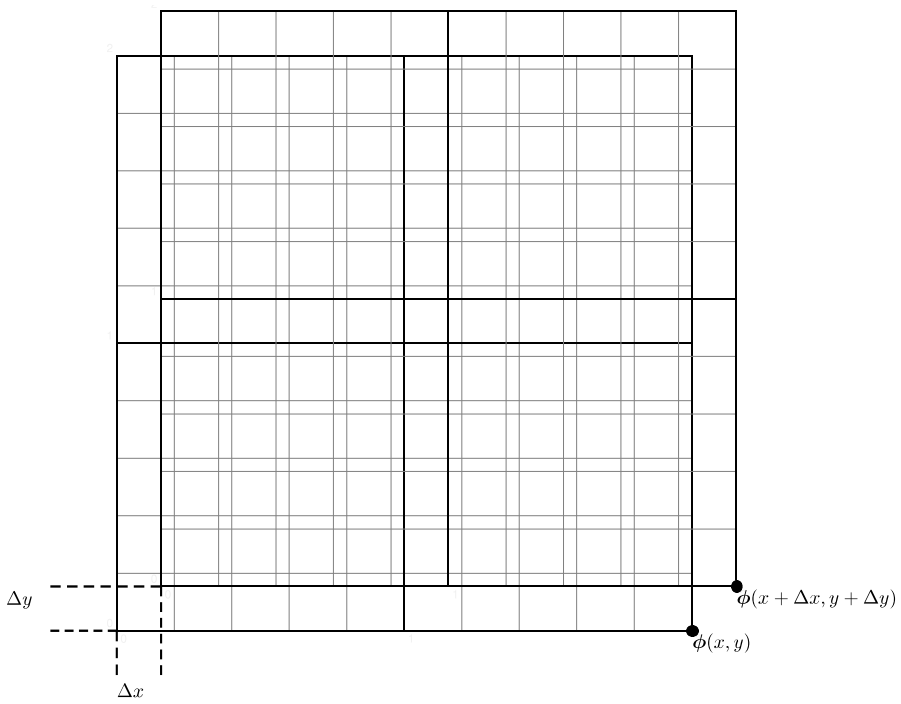

Figure 5.1: Initial and Phase Shifted Scalar Data Set $\boldsymbol{\phi}$

$$Im(f_x(\phi(x + \Delta x, y, z))_i) = r_i sin(k_i \Delta x) + j_i cos(k_i \Delta x) \tag{5.1c}$$

$$\boldsymbol{\phi} = IFFTX(f_x(\boldsymbol{\phi})) \tag{5.1d}$$

Figure 5.2 shows the steps of the algorithm that uses the above equations. The sampling used with this method can be classified as quasi random sampling, for a discussion see Press et al. [53, pp. 404-409]. Figure 5.3 graphically shows sample convergence rates for Monte Carlo integration with random and quasi-random sampling with hard and soft boundaries, which relates to the existence of a pattern between the surface and sampling pattern. Thus, the figure 5.3 lends theoretical justification to our algorithm.

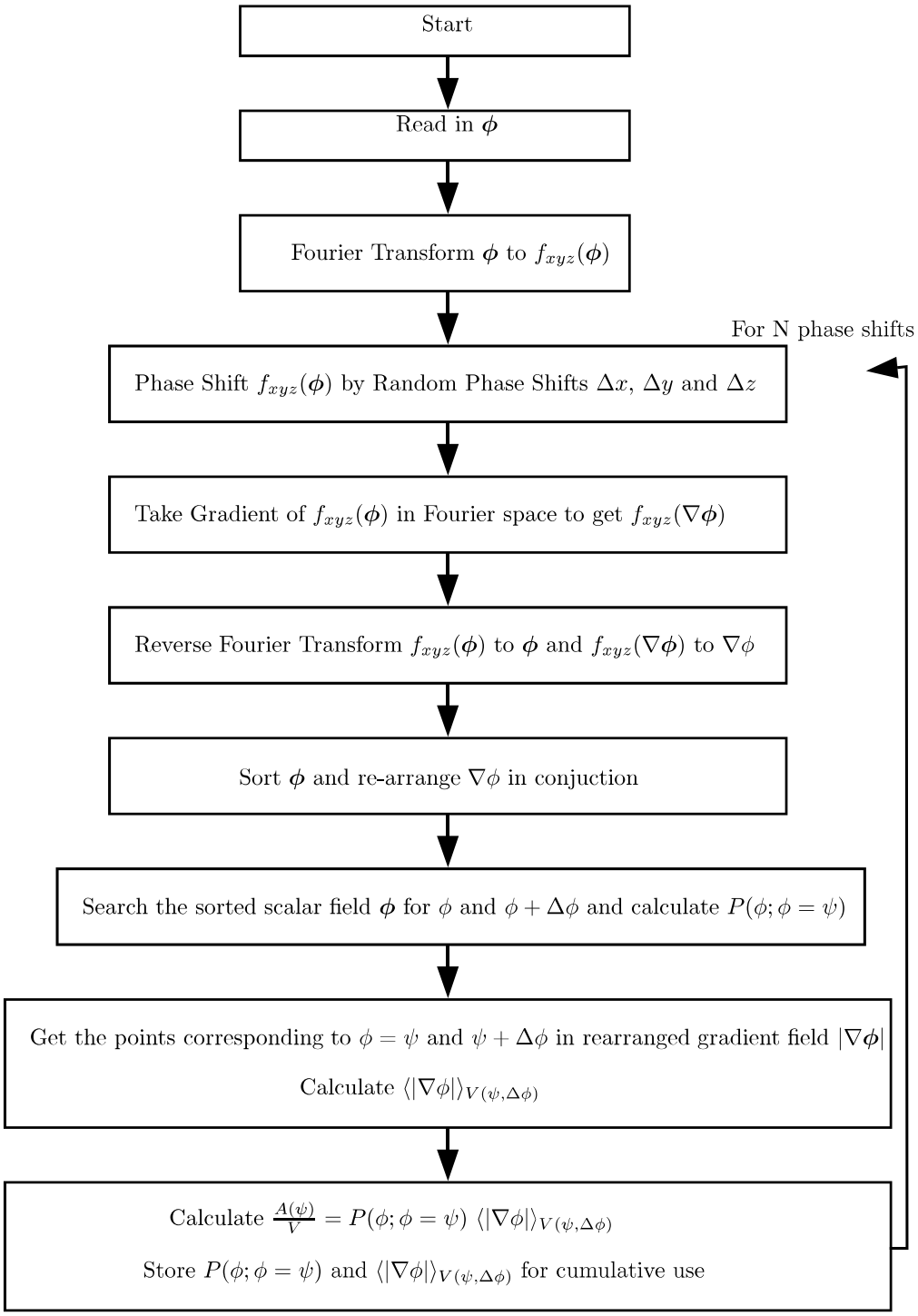

Figure 5.2: Algorithm for Monte Carlo Method

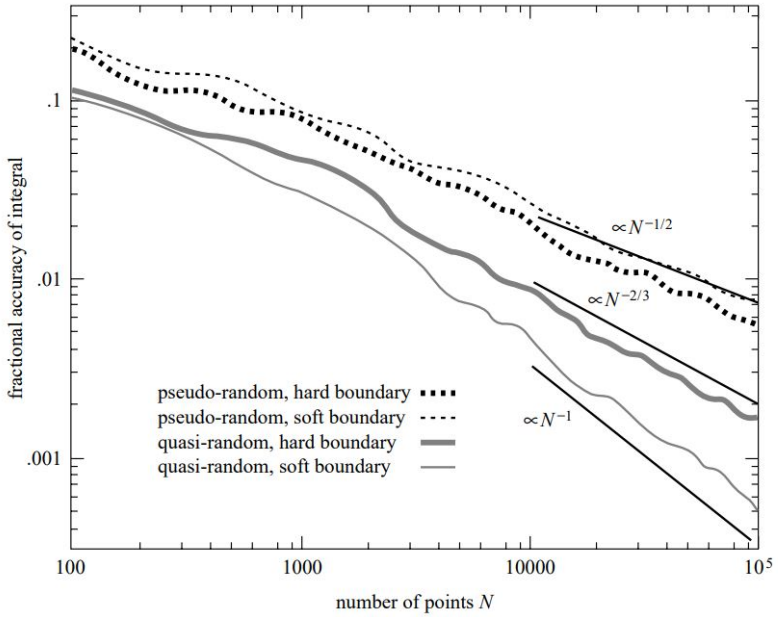

Figure 5.3: Convergence rates for various sampling methods (Numerical Recipes, The Art of Scientific Computing)

5.2 Test Cases

In this chapter, we present four test cases that have different test functions as ϕ, with the dual purpose of verifying our method, presented in 3 and our code, presented in 5.1. We have attempted to choose cases that helped us test the code for accuracy and precision.

5.2.1 Case 1: $\phi(x, y, z) = cos(x) + cos(y) + cos(z)$

This function was chosen as a test case because of its periodic and symmetric nature in three dimensions. Even though an analytical integral for the area of an iso-surface of this field does not exist, it was chosen for its use in debugging the code. The code was tested with three grid sizes for input data sets of ϕ namely 128^3, 256^3 and 512^3 to confirm that the quantities of interest converge to the same value, given sufficient phase shift interpolations. Figure 5.4 shows that the method converges for any grid size with sufficient phase shift sampling. This implies that if the grid in the

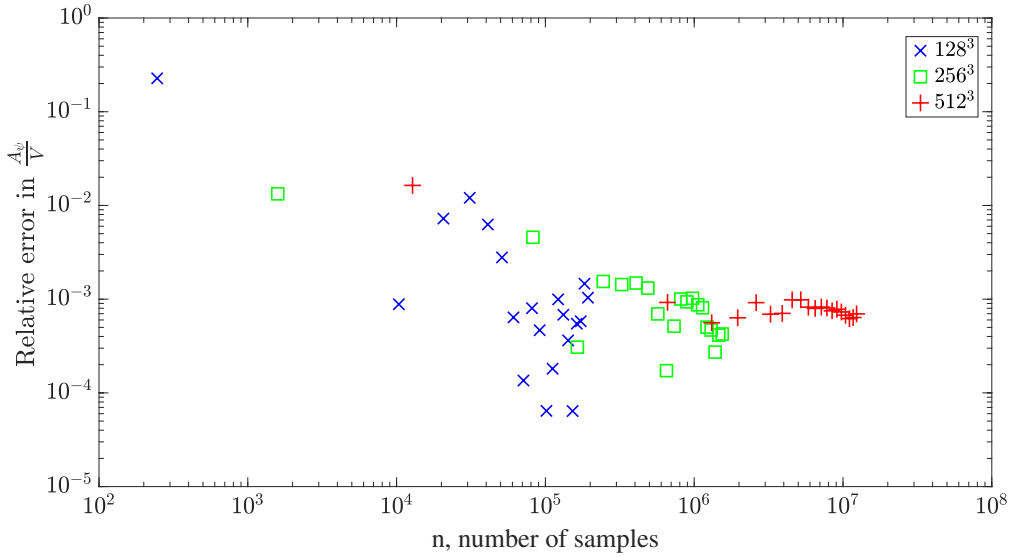

Figure 5.4: Convergence of area ratio plotted with number of points on the layer, with three grid sizes for scalar value $\psi = 2$ and a bin width $\Delta\phi = 0.001$. It can be seen that all three calculations converge to the same result, which means that accuracy does not depend upon starting grid size.

data set resolves ϕ sufficiently, the size of the data set does not affect the accuracy of our calculation. Figure 5.5 shows the convergence of the area ratio calculation with fixed grid size and different bin widths. We see that the calculation converges for any bin width, or in other words, for arbitrarily high fidelity of desired iso-surface. Another observation is that the accuracy of calculation increases with smaller bin widths which provides a higher fidelity on the iso-surface. The "exact" value of the area used to calculate relative error for 5.4 and 5.5 was taken from the marching cubes algorithm provided by MATLAB(2018a).

In order to test with more grid points, we tested the method with another test function, in two dimensions.

5.2.2 Case 2: $\phi(x, y, z) = cos(x) + y$

Consider the test scalar field $\phi(x, y, z) = cos(x) + y$ over the domain $\boldsymbol{x} \in [-\pi, \pi)$ and range $\phi(\boldsymbol{x}) \in [-1 - \pi, 1 + \pi]$This function was chosen as a test case because it is

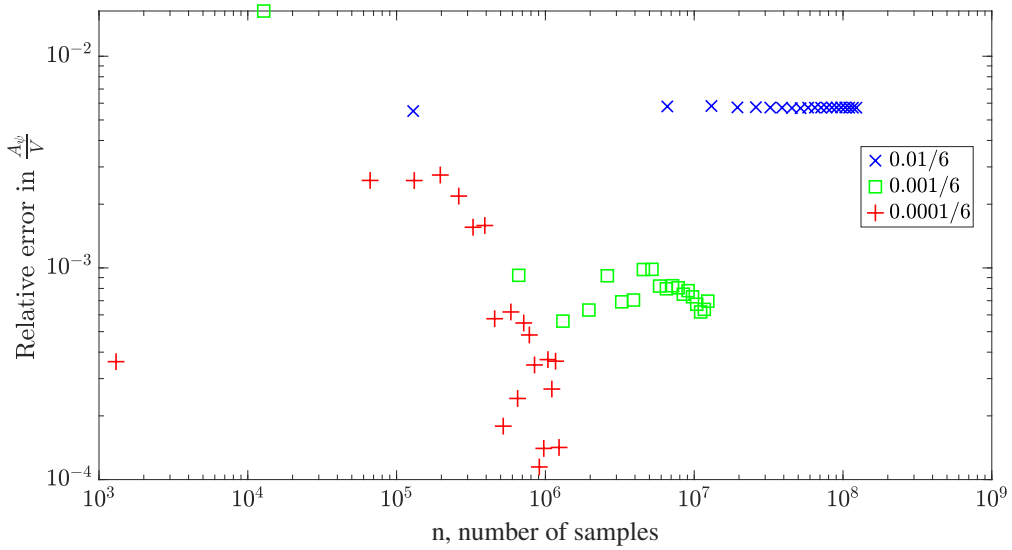

Figure 5.5: Convergence of area ratio plotted with number of points on the layer, for scalar value $\psi = 0$ with three bin widths 0.01, 0.001 and 0.0001. It can be seen that the area ratio converges for arbitrary bin width, given sufficient number of samples

two dimensional in complexity(more sampling) and has an analytical integral solution for the area. The code was tested with with a grid of 100 points in each direction, with a 1000 random phase shifts for sampling. This kind of sampling is similar to the quasi-random sampling referred to in Press et al. [53]. Figure 5.6 shows that the area ratio calculation converges as the number of samples is increased.

5.2.3 Case 3: $\phi(x, y, z) = cos(x)$

As a further check, the sample scalar field used in Chapter 4 was substituted by $\phi(x, y, z) = cos(x)$ over the domain $\boldsymbol{x} \in [-\pi, \pi)$ and range $\phi(\boldsymbol{x}) \in [-1, 1]$ with N points along each axis and total size N^3. If ψ is a constant such that $\psi \in [-1, 1]$, the iso-surface of $\phi = \psi$ is given by the roots of the equation $\phi(\boldsymbol{x}) = \psi$. In this sample case, the equation is $cos(x) = \psi$ which has two roots x_1 and x_2 in general(except for the boundary values). The Iso-surface can described as $\{\boldsymbol{x}(x, y, z)|x = x_1 \; or \; x_2, y \in [-\pi, \pi), z \in [-\pi, \pi)\}$ which describes a set of two planes in the field. The area ratio $A(\psi)$ corresponding to the iso-surface given $\phi = \psi$ can be easily calculated.

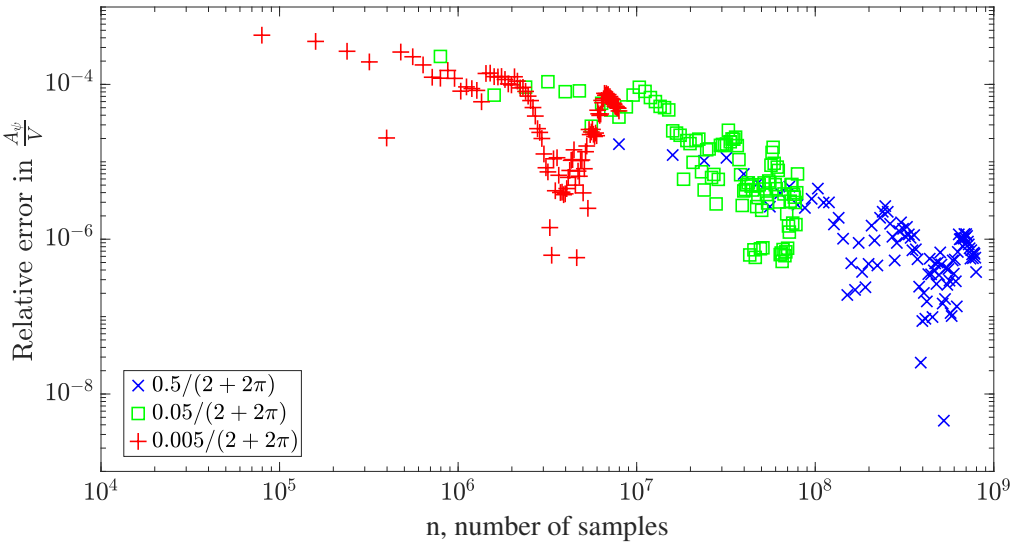

Figure 5.6: Convergence of area ratio plotted with number of points on the layer, with three bin widths 0.1, 0.01 and 0.001. The area ratio converges to the analytical result.

$$A(\psi) = 2 \times \frac{(2\pi)^2}{(2\pi)^3} = \frac{1}{\pi} = 0.31830988618$$

The convergence of this calculation was tested with 800,000 phase shifts/interpolations for $\psi = 0.5$ and $\Delta\phi = 0.01$ and relative error was calculated for the exact area ratio $A(\psi)$. Figure 5.7 shows that the area ratio calculation can converge to arbitrarily high accuracy. Figure 5.8 shows the same on a log-log plot which supports the $N^{-2/3}$ convergence of Monte Carlo Integration with quasi random sampling and hard boundary, as shown in figure 5.3. In this case, the presence of a planar surface is the reason for a hard boundary as it reduces the three dimensional sampling to an effective one dimensional sampling.

5.2.4 Case 4: $\phi(x, y, z) = x^2 + y^2 + 2z$

To illustrate the evaluation of iso-surface area via (3.6), we consider the scalar field $\phi(\vec{x}) = x^2 + y^2 + 2z$ in the volume $x \in [0, 2], y \in [0, 2], z \in [0, 2]$. In this test case

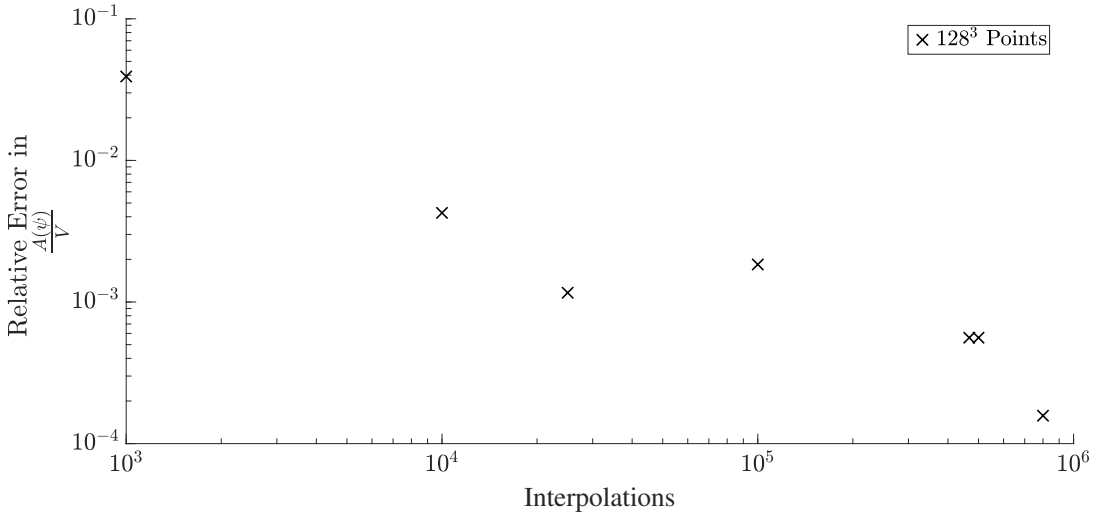

Figure 5.7: Convergence of $A(\psi)$ with 800,000 interpolations

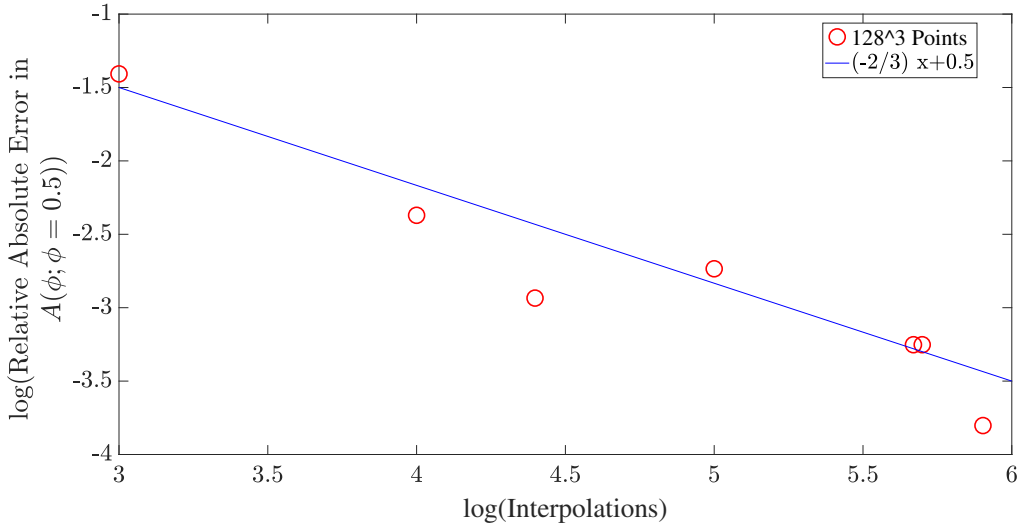

Figure 5.8: Convergence of $A(\psi)$ with 800,000 interpolations, $-2/3$ slope

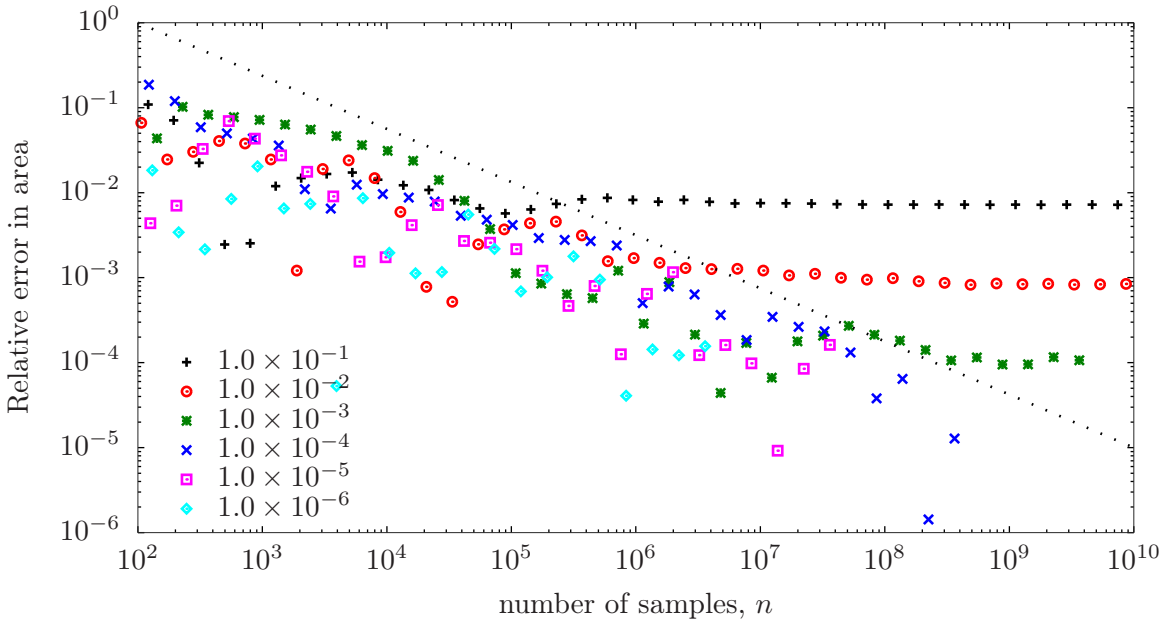

Figure 5.9: Relative error in the estime for the area of the iso-surface in the simple example with $\psi = 4$.

ϕ is known exactly so that the area of any iso-surface and the mean gradient on the surface are known analytically.

Figure 5.9 shows the results of sampling ϕ many times and accumulating the statistics for various values of $\Delta\phi$. It is observed that the relative area in the area estimate converges with $n^{-1/2}$ with n the number of samples, which is the expected convergence rate for pseudo-random sampling. For each $\Delta\phi$ a minimum relative error is reached given enough samples. The term "relative error" is not strictly correct because the method has arrived at the correct areas for layers of finite thickness. For this simple case, the area of the surface with $\phi = \psi + \Delta\psi$ will always be larger than the area of the iso-surface $\phi = \psi$. For example, the relative difference between the analytically determined areas of the surfaces with $\psi = 4$ and 4.01 is approximately 0.001, which is consistent with figure 5.9. We conclude from this simple case that the method can be used to measure iso-surface areas with arbitrary accuracy given a sufficiently small $\Delta\psi$ and sufficient samples. One caveat is that care must be used

in accumulating the statistics, such as by using 128-bit floating point arithmetic and, perhaps, the method of Kahan [28] to reduce roundoff errors. A second caveat is that pseudo-random sequences have limitations which are beyond the scope here but are discussed in Vadhan et al. [e.g. 71].

We verified our method and code using several cases, and have provided the most significant of them in this chapter. In the next chapter, we apply this method to Direct Numerical Simulation (DNS) data and calculate iso-surface area statistics for those cases.

5.2.5 Case 5: Potential Modifications to the Integral Method

The formulation for the isosurface area provided in 3 and the error analysis provided in B creates a strong foundation for analyzing integral methods. This enables us to explore two modifications to this method that might be useful in situations when the calculation is severely limited by data and/or computing capacity.

In B, we show that the quantity $V(\psi, \Delta\phi)/V\ \Delta\phi$ in 3.6 can be interpreted as a first order approximation to the p.d.f $P(\phi; \phi = \psi)$. Epanechnikov [20] explores improvements to the process of calculating p.d.f.'s by histograms, seeking to minimize global error in the histogram estimate of the p.d.f. Schmidt and Bedford [59] use this concept for a different application related to spray simulations and provide a simplified and practical explanation of it's use. Epanechnikov [20, Table 1] shows a list of "kernels" or bin widths with positions which are designed to minimize the global error in the histogram. Our use of left sided bins is a special case of these kernels. We are not seeking to minimize global error, as we are not trying to calculate the entire p.d.f., only a first order approximation to it for certain isovalues. The idea of "kernels" have a simple analog for minimizing local error which is related to numerical integration. Left handed bins correspond to the rectangle rule for numerical integration which has first order convergence with reducing layer thickness. The Epanechnikov Kernel

has centered bins with a width specified by Epanechnikov [20, Table 1, Row 1]. The analog of this idea for local estimates is the midpoint rule for the integral in 3.4, which should give us second order convergence with reducing layer thickness. This requires bins centered on $\phi = \psi$. We tested this for $\phi(x, y) = y - x^2$ on the domain $x \in [0, 1], y \in [0, 1]$ and observed the predicted second order convergence. 5.10 compares the two different approaches with bin positioning. However, the definition of the p.d.f. used in turbulence literature requires the histogram to be created with the left handed rectangle rule. In order to be consistent with the literature, we performed the calculations with a first order method for all of our plots.

The other modification explored by Epanechnikov [20, Equation 12] is the relation between the number of available samples and the bin width that will provide an optimum estimate to the p.d.f., similar to the approach of Freedman and Diaconis [22] discussed in 4, the difference being that the author attempts to provide a non-parametric estimate without assuming an underlying distribution to the field. Their purpose is to reduce the global error, which is not our focus here. For univariate p.d.f. estimation, the equation simplifies to

$$h_0(N) = \frac{c}{N^{1/5}} \tag{5.2}$$

where $h_0(N)$ is the optimum bin width as a function the total number of samples N and c is a constant that depends on the field ϕ. The field given in this section was sampled at different number of points, and for each of them, an optimum bin width was determined by the process of trying a range of bin widths and finding a minima for the relative error. The optimum bin width was plotted against the corresponding total number of samples and has been shown in 5.11.

The constant c cannot be obtained analytically as it involves a calculation that requires the analytical p.d.f. It can be seen from 5.11 that it would be difficult to determine the constant c numerically due to the amount of scatter, which can be

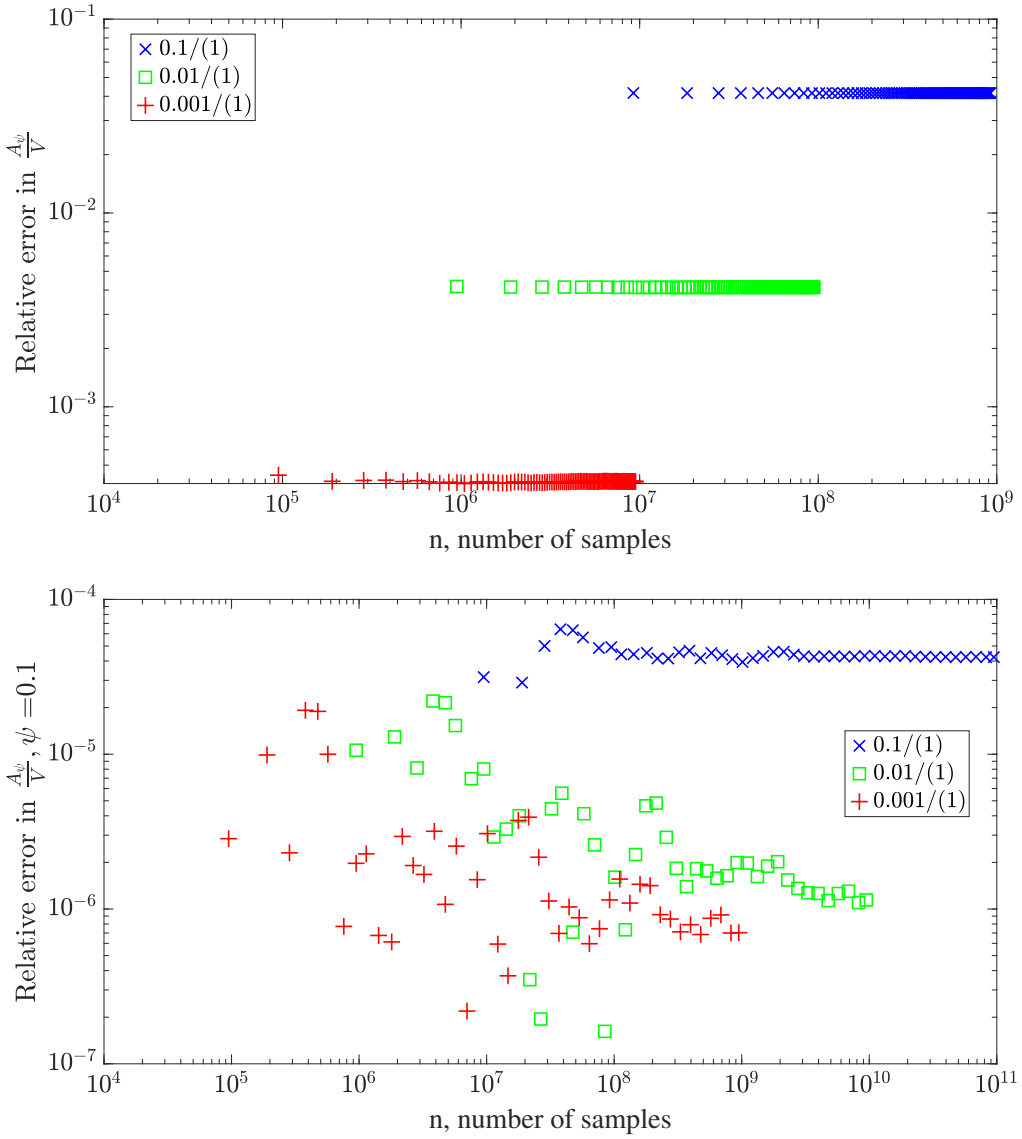

Figure 5.10: First order convergence of area ratio with bin width using left sided bins (panel (a)) and Second order convergence of area ratio with bin width using centered bins (panel (b)). This agrees with the expected convergence for the rectangle and midpoint rules for numerical integration.

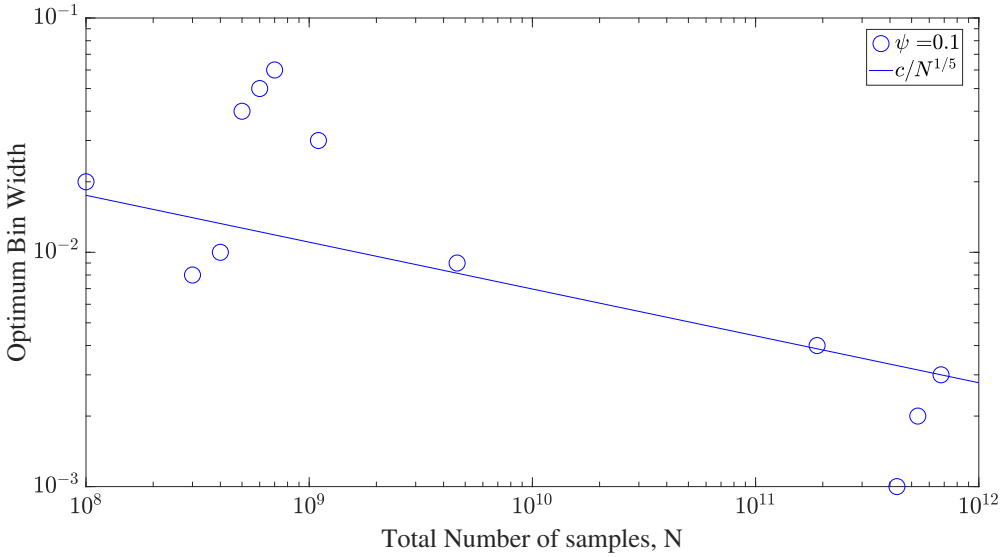

Figure 5.11: Optimum bin width determined by testing for different number of samples. The line represents the optimum bin width as predicted by (5.2) adjusted to the rightmost point. It can be seen that the calculations agree broadly with the prediction, but there is a certain amount of scatter.

attributed to a number of reasons like effects of this particular distribution on the calculation, our ability to numerically extract the true minima (limited by the number of bins we can try) and the effects of an additional multiplier function, which is the average gradient. Also, this method is designed for minimizing global least square error in p.d.f.'s, not the local error. Another challenge that we mentioned before is that our calculation involves a second function, namely, the average gradient, the convergence of which has not been addressed by the equation. Thus, the optimum bin width calculation given in (5.2) cannot be used directly for the area calculation problem.

CHAPTER 6

DIRECT NUMERICAL SIMULATIONS

The simulated flows are the solution to the Navier-Stokes equations and passive scalar transport equation in three spatial dimensions:

$$\nabla \cdot \vec{u} = 0 \tag{6.1a}$$

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = -\nabla p + \nu \nabla^2 \vec{u} + \vec{b} \tag{6.1b}$$

$$\frac{\partial \phi}{\partial t} + \vec{u} \cdot \nabla \phi = -\vec{u} \cdot \nabla \Phi + D \nabla^2 \phi \ . \tag{6.1c}$$

The velocity vector is $\vec{u} = [u, v, w]$ with kinematic viscosity ν, the pressure p has been divided through by the (constant) density, and $\vec{b}$ is a time-varying force applied to maintain the flow statistically stationary. The passive scalar is decomposed into $\Phi + \phi$, with time-invariant Φ having a uniform gradient in the direction of w, and being uniform in the other spatial directions; ϕ denotes the fluctuations relative to Φ with molecular diffusivity D.

Spatial discretization is via Fourier series truncated according to the 2/3 rule to remove all effects of aliasing. The advection terms are computed in real space in vorticity form for momentum and in convective and conservation forms on alternating time steps for the scalars. The equations are advanced in time using a third order accurate fractional step method for the advection and pressure gradient terms while the diffusion terms are integrated exactly in Fourier space.

The velocity fields are forced isotropically so that the three-dimensional kinetic energy spectrum at length scales larger than the integral length scale matches Pope's

model spectrum with his $p_0 = 2$ and his $c_L = 6.78$ [49, (equation 6.247)]. Each time step, a second-order ordinary differential equation is advanced in time for each shell of the shell-averaged spectrum to determine how much energy to add to that shell. The energy is distributed randomly, but consistent with continuity, across all the wave numbers composing the shell. The solution to these differential equations and the random distributions produces the Fourier-space equivalent of $\vec{b}$. This approach is introduced by Overholt and Pope [41] to efficiently converge a simulation to a target spectrum; Rao and de Bruyn Kops [55] reviews a variety of forcing techniques for the type of simulation used for this research.

Five velocity fields are considered for this research, with each convecting four scalar fields with differing Schmidt numbers $Sc = \nu/D$. The velocity fields are denoted R1 through R5 with R1, R2, and R3 being statistically equivalent to those reported by Almalkie and de Bruyn Kops [1] and R4 equivalent to the isotropic homogeneous case used as a reference in Almalkie and de Bruyn Kops [2], de Bruyn Kops [11], Portwood et al. [52]; velocity field R4 with $Sc = 0.7$ and 1.0 is used by Muschinski and de Bruyn Kops [35] to study models for scalar energy spectra.

Many of the simulations were run with multiple resolutions for the numerical studies presented in 7. For evaluating surface areas as a function of Reynolds and Schmidt numbers, the topic of 8, the statistics of the simulations with highest resolution are of interest. These are listed in table 6.1 with Re_L the Reynolds number based on the r.m.s. velocity and the integral length scale L, Re_λ the Reynolds number based on the r.m.s. velocity and the Taylor microscale and Pe_λ the corresponding Peclet number.

L_k, L_b, and L_c are the Kolmogorov, Batchelor, and Oboukhov-Corrsin length scales, respectively,

$$L_k = \frac{\nu^3}{\epsilon}^{1/4}, \quad L_b = \frac{\nu D^2}{\epsilon}^{1/4}, \quad L_c = \frac{D^3}{\epsilon}^{1/4} \tag{6.2}$$

with corresponding times scales

	R1				R2				R3				R4				R5			
Re_L	310				734				1853				4790				12150			
Re_λ	97.8				154				245				400				633			
$\mathcal{L}/L$	5.18				5.71				5.68				5.56				5.56			
$\kappa_{max}L_k$	26.84				12.90				6.48				9.39				5.65			
N	4096				4096				8800				11760				14256			
Sc	0.1	0.7	1.0	7.0	0.1	0.7	1.0	7.0	0.1	0.7	1.0	7.0	0.1	0.7	1.0	7.0	0.1	0.7	1.0	7.0
Pe_λ	9.8	68	98	685	15.4	108	154	1076	24.5	171	245	1712	40.0	280	400	2800	63.3	443	633	4429
$Ł_k/\Delta x$	12.8	12.8	12.8	12.8	6.2	6.2	6.2	6.2	4.5	4.5	4.5	6.6	2.2	2.2	2.2	4.5	1.5	1.5	1.5	2.7
$Ł_c/\Delta x$	72.1	16.7	12.8	3.0	34.6	8.0	6.2	1.4	25.5	5.9	4.5	1.5	12.5	2.9	2.2	1.0	8.7	2.0	1.5	0.6
$Ł_b/\Delta x$	40.5	15.3	12.8	4.8	19.5	7.4	6.2	2.3	14.3	5.4	4.5	2.5	7.0	2.7	2.2	1.7	4.9	1.9	1.5	1.0
$\tau_k/\Delta t$	1376	2001	2580	2043	4751	801	1165	1501	755	3018	509	740	613	486	1940	327	295	380	725	1202
$\tau_c/\Delta t$	4352	2392	2580	772	15023	957	1165	567	2389	3607	509	280	1939	580	1940	124	932	454	725	454

Table 6.1: Simulation parameters and statistics for the highest resolution of each case. N and $\kappa_{max}L_k$ are shown for each velocity field at it highest resolution ($Sc = 7$); lower resolution is used for the cases with lower Sc in simulations R3, R4 and R5.

$$\tau_k = \frac{\nu}{\epsilon}^{1/2}, \quad \tau_c = \frac{D}{\epsilon}^{1/2} . \tag{6.3}$$

The maximum wave number after dealiasing is κ_{max}, the grid spacing is Δx, and the time step size is Δt. The domain size is $\mathcal{L}$ in each spatial direction discretised by N is the number of grid points.

CHAPTER 7

ADDITIONAL NUMERICAL CONSIDERATIONS

Before proceeding to evaluating the iso-surface areas in the DNS as a function of Reynolds and Schmidt numbers in 8, it is important to apply the tests from 5.2 to understand the accuracy to which the areas can be evaluated from the DNS. It is also important to understand the extent to which the spatial and temporal resolution of the DNS may effect the area calculations. These are the topics of this chapter. We do not review the numerical errors in Fourier spectral methods because they are discussed so extensively in the literature and instead refer the reader to, e.g., de Bruyn Kops and Riley [13, c.f. §3].

Fourier spectral simulations represent the scalar fields as infinite series with all but a few of the series coefficients identically zero. This is in contrast to, say, a laboratory time series which has been Fourier transformed and represented by a truncated series. This characteristic of the current simulations allows us to sample the scalar fields at an arbitrary number of locations via phase shifting so that each sample has the same accuracy as the simulation as a whole and no interpolation error is introduced. Also in Fourier spectral simulations, $\nabla\phi$ is known with the same accuracy as ϕ because the derivative of each term in the series expansion is known analytically.

The sampling process begins with the scalar field in Fourier space. A random phase shift with wave number vector $\vec{\kappa}$ is applied, the field is transformed to real space, and all of the points in the resulting field are used to accumulate data for evaluating (3.6). The process is repeated as many times as desired. The result is a Monte Carlo method in which for every N^3 samples of the field only one random vector

κ is chosen. Because the samples for a single value $\vec{\kappa}$ are uniformly distributed across the entire simulation domain, this method is a type of "quasi-random" sampling (5.1).

The spatial and temporal resolution requirements in DNS have been continually refined over the last three decades as computers have gotten more powerful so as to enable simulations with higher internal intermittency. The large-scale requirement for decaying isotropic homogeneous turbulence (IHT) was established by experimentation to be about 20 times the integral length scale ℓ by de Bruyn Kops and Riley [12] and recently verified by de Bruyn Kops and Riley [14]. In simulations forced to be statistically stationary, such as those used for the current research, no large-scale resolution requirement has been established to our knowledge, and we have used domains that are approximately 5ℓ because this resolution has been shown to produce structure functions and spectra that match theoretical predictions in the inertial and dissipation ranges [1].

Small scale resolution requirements for DNS were first established by Eswaran and Pope [21], who showed that $\kappa_{max} L_k > 1$ is required for the scalar variance to decay as expected from theory in decaying IHT. Since then, the dependency of small scale statistics on Reynolds number has been analyzed in several studies [23, 24, 25, 26, 76, 77] and the effects on intermittency of insufficient small-scale resolution in DNS investigated [18, 62, 63, 72, 74]. Recently, Yeung et al. [75] report on not only the spatial resolution required to resolve small-scale intermittency but also the temporal resolution for $Re_\lambda = 390$ and 650, which are comparable to our cases R4 and R5. Because of our need to resolve $Sc = 7$, our spatial resolutions are finer than those shown to be sufficient in the paper just referenced. Therefore, when we satisfy the Courant number condition established by Yeung et al. our timestep is smaller than theirs, which gives us confidence that the time scale of intermittency is resolved. The time step in each simulation is given at the bottom of table 6.1 relative to the Corrsin

and Kolmogorov time scales from which it is apparent that these time scales are very well resolved in the simulations.

The foregoing gives us confidence that all the scalar and velocity fields are well resolved and so we move to the question of the effect of bin width $\Delta\phi$ used in numerically evaluating (3.6). For the simple example illustrated by figure 5.9, the error in the iso-surface area reduces unbounded and proportionally to $\Delta\phi$. Recall that the exact area in that case is known. In the DNS fields, though, the exact area is not known so that the error in the area must be based on the best available area estimate, e.g., that computed with the scalar field at the highest resolution available and with the smallest practicable bin width. Also, we can surmise that, even with careful attention to mitigating them with the method of Kahan [28], roundoff errors will limit the accuracy of the numerical evaluation of (3.6).

Our test case for numerical studies is R3 because the Reynolds number is high enough for there to be significant intermittency yet we have resolved it with much finer spatial resolution than required per Yeung et al. [75]. Pr = 7 is considered since it has the smallest diffusive length scale. The relative errors for the area of a single isosurface ψ computed using various bin widths is shown as figure 7.1. The dimensionless bin width is

$$\widetilde{\Delta\phi} = \frac{\Delta\phi}{L_r \left\langle |\nabla\phi| \right\rangle_{\boldsymbol{x}_\psi}} \tag{7.1}$$

with $\Delta\phi$ the dimensional bin size used when computing $P(\phi; \phi = \psi)$ and L_r some resolution length scale appropriate for the scalar. In this thesis we set $L_r = \Delta x$ because, in our simulations, $\Delta x < L_b$ for $Pr \geq 1$ and $\Delta x \leq L_c$ for $Pr \leq 1$.

The first thing to note from the figure 7.1(a) is that the leftmost symbol in each set is the number of samples n in a layer of thickness $\widetilde{\Delta\phi}$ before phase shifting, e.g., when $\widehat{\Delta\phi} = 7.0 \times 10^{-5}$ approximately 10^5 points are in $[\phi, \psi + \Delta\phi]$. Next it is noted that $\widetilde{\Delta\phi} = 8.1 \times 10^{-2}$, the minimum relative error is approximately 10^{-3} and does not decrease with more sampling. For smaller $\widetilde{\Delta\phi}$, the iso-surface area estimates converge

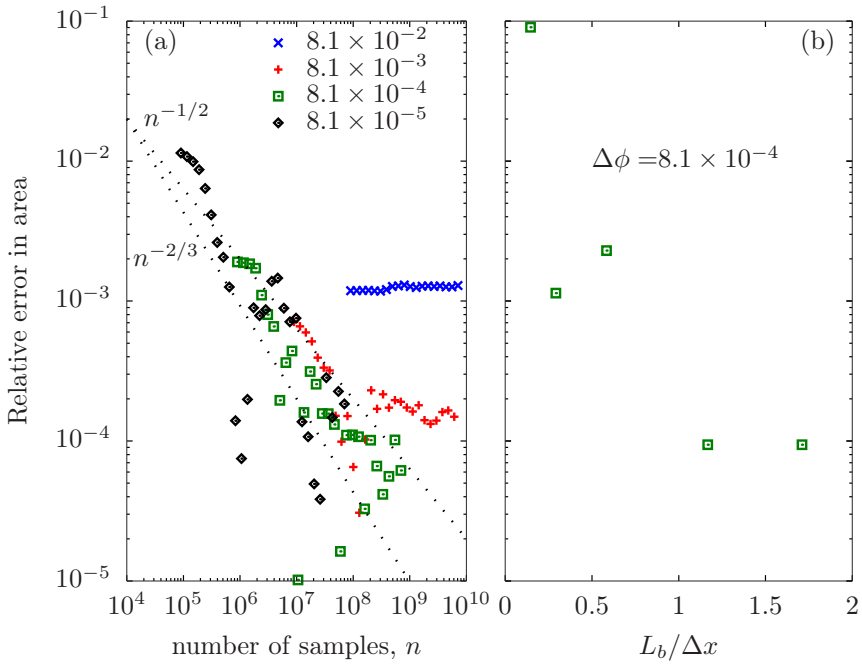

Figure 7.1: Error in the area of one iso-surface ψ in case R3, $Pr = 7$, for various nondimensional bin widths $\widetilde{\Delta\phi}$ (panel (a)) and grid resolutions (panel (b)). The dotted line is a reference slope.

with $n^{-1/2}$ as expected (4) before reaching some minimum relative error. For example, when $\widetilde{\Delta\phi} = 8.1 \times 10^{-3}$ then the minimum relative error is approximately 1×10^{-4} meaning that the iso-surface area estimate converges to a value different from the "exact" value taken to be the estimate when $\widetilde{\Delta\phi} = 8.1 \times 10^{-5}$.

From figure 7.1(a) we conclude that the error in the area estimate can be made arbitrarily small given a small enough bin width and a sufficient number of samples produced by phase shifting. This conclusion is conditioned on the assumption that the simulations are sufficiently resolved at the small scale. We have already concluded that we expect them to be, based on the requirements determined by Yeung et al. [75] but we verify it one more time with figure 7.1b. The "exact" area for this plot is from a simulation with $L_b/\Delta x = 2.5$ that is filtered with a spectral cutoff filter to produce the fields with lower values of L_b/Δ used to make the figure. It is observed that the error in the area estimates converges to its minimum value when $L_b/\Delta x = 1$.

Note from figure 7.1(a) that this minimum value, approximately 1×10^{-4} is set by the bin width, not the resolution of the simulation.

CHAPTER 8

ISOSURFACE AREA FOR DNS OF ISOTROPIC HOMOGENEOUS TURBULENCE

The time rate of change of the area of a non-material iso-surface is determined by the balance between stretching tangential to the surface and diffusion normal to the surface. A detailed derivation is included in Dopazo et al. [19] for the case of a chemically reacting flow. For our purpose here we follow their notation and consider an infinitesimal piece of an iso-surface having area $\mathcal{S}$ with normal vector n_i and strain rate S_{ij} so that the strain rate tangential to the surface is $a_T = (\delta_{ij} - n_i n_j)S_{ij}$. A point on the iso-surface moves by diffusion, relative to the speed at which the surface is advected, with speed $S_d = -D\nabla^2\phi/\nabla\phi$. Therefore the fractional change in the area of this piece of the isosurface is [c.f. 19, equation 25]

$$\frac{1}{\mathcal{S}}\frac{\partial \mathcal{S}}{\partial t} = a_T + S_d \frac{\partial n_i}{\partial x_i} \ . \tag{8.1}$$

For the statistically stationary flows, the integral of (8.1) over an entire iso-surface is zero meaning that the two terms on the right hand side come into equilibrium. The central physics question motivating this thesis is how the equilibrium surface area $A(\psi)$ is affected by Reynolds and Schmidt number. This question has been addressed in the literature [7, 40, 63]. Schumacher and Sreenivasan [61] consider DNS of a passive scalar with a mean gradient in isotropic homogeneous turbulence with $10 \leq Re_\lambda \leq 42$ and $2 \leq Sc \leq 32$ and observe that the area-volume ratio increases with both Sc and Re_λ but that a power law is not observed. They further note that power law behaviour is not expected because, at the Reynolds and Schmidt numbers considered,

neither the inertial-convective nor the viscous-convective subranges exist. Catrakis et al. [7] measured area-volume properties of fluid interfaces in a laboratory water jet with jet Reynolds number $\sim 20,000$ and $Sc \sim 2000$ and report that, while large and small length scales contribute to the area-volume ratio, the dominant effect at is at the small scales. They conclude that self-similarity at the small scales can be expected to enable extrapolation of area-volume behaviour to higher Reynolds number. O'Neill and Soria [40] also consider DNS of isotropic homogeneous turbulence with a mean scalar gradient with $Sc = 0.25$ and 5.0 and $Re_\lambda = 41$, 42, and 77. They observe that increasing either Re_λ or Sc results in the size of the isoscalar surfaces decreasing. They show plots of the iso-level surfaces of scalar fluctuations at varying Reynolds number with constant Schmidt number and vice versa and list causes for an observed, unexpected decrease in area.

In the literature just cited, the metric for isosurface area is the area-volume ratio. Our simulations are computed in periodic domains, however, so that the scalar can be convected out of the domain to a periodic domain above or below. Given this configuration, we find it to be more clear to take advantage of homogeneity in the flows and consider the dimensionless ratio of the wrinkled surface to the initial surface area, which is that of a plane normal to the mean scalar gradient. The wrinkled surface area includes that of any portions of the surface advected across the periodic boundaries. To be precise, we compute the volume of a very thin wrinkled layer with $\widetilde{\Delta\phi} = 1 \times 10^{-4}$ and relate it to the isosurface area via (3.6). The results are in figure 8.1. For each case, the areas are measured for 20 iso-values, spanning the range of ϕ in the simulation. It is noted that there is more scatter in some cases than in others, and this appears to be due to limited large-scale resolution so that large structures occur which are resolved in space but not resolved statistically.

The provisional conclusion drawn from figure 8.1 is that the area increases with $Pe_\lambda^{1/2}$ for the entire range of Péclet numbers accessible with this simulation technique.

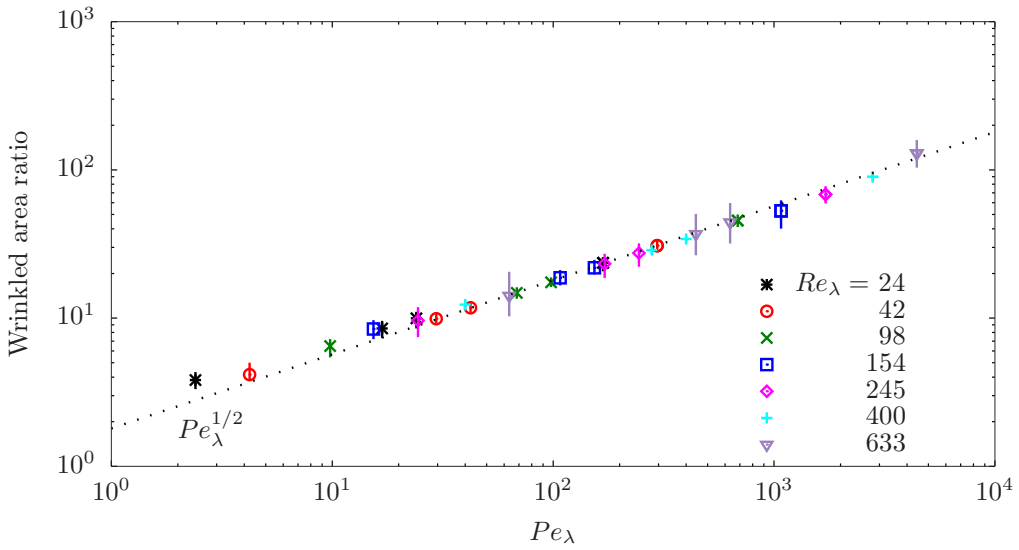

Figure 8.1: The iso-surface areas normalized by the area of a horizontal plane in the simulations. The symbols mark the average for 2 iso-values and the bars indicate the range from the largest to smallest areas. The dotted line is a reference line, not a fitted line.

Our results are in broad agreement with the conclusion of Catrakis et al. [7] that small-scale similarity is expected; note that the current flow configuration does not allow us to compare with their results regarding variations in large-scale features of the flow. Our results are also broadly consistent with the observation of Schumacher and Sreenivasan [61] that the area increases with increasing Re_λ and Sc.

We end with a brief discussion as to why the isosurface area might increase as $Pe_\lambda^{1/2}$. Sreenivasan [66, equation 10] mentions that at high Péclet number $Pe_\lambda > O(100)$, the scalar variance dissipation rate given by $\chi = 2D\nabla\phi \cdot \nabla\phi$ is largely independent of Pe_λ. Now, at constant Re_λ, the Péclet number and the diffusivity D are inversely proportional. Therefore, for a constant χ, $\nabla\phi \cdot \nabla\phi$ is proportional to Pe_λ. Thus, $\nabla\phi$ is proportional to ${Pe_\lambda}^{1/2}$. In general, isosurface areas increase with increasing gradients. Thus, we postulated that the isosurface area increases as ${Pe_\lambda}^{1/2}$.

CHAPTER 9

CONCLUSIONS

The area of a scalar isosurface can be written exactly in terms of the limit of the volume of a thin layer as the layer thickness goes to zero, which is shown in the development of (3.6). In 3 and B, we provide a robust foundation for the analysis and development of integral methods for calculating isosurface areas and provide an error analysis for the same. Numerically evaluating (3.6) to arbitrarily small relative error in DNS via a Monte-Carlo method requires being able to accurately evaluate the scalar field at arbitrary locations, which can be done in the case of pseudospectral simulations. We demonstrate the evaluation of the surface areas for five analytical test cases and then with DNS data and observe the expected convergence behaviours as the thickness of the layer is reduced and the number of samples increases. We also briefly look at ways to adjust the method for cases with limited data and/or limited computing capacity. For the DNS, it is observed that with sufficient spatial resolution of the simulation and a sufficiently thin layer width that the relative error in the area of an isosurface can be reduced to 1×10^{-4} or smaller. In the process of determining this, we verify the small-scale resolution criterion recently provided by Yeung et al. [75].

Next the methodology is applied to 20 DNS cases of statistically stationary isotropic homogeneous turbulence with a mean scalar gradient, $0.1 \leq Sc \leq 7$ and $97.8 \leq Re_\lambda \leq 633$. It is observed that the layer area is very nearly proportional to ${Pe_\lambda}^{1/2}$ over the entire range $9.8 \leq Pe_\lambda \leq 4429$, which is broadly consistent with the literature.

APPENDIX A

WHY WE NEED BINS LIMITS FROM x TO $x + \delta x$

Let $p(x)$ be the probability density function of continuous random variable x. Let $P(x)$ be the cumulative distribution function of x. Then, by definition of probability density function, it is given by A.1a. The cumulative distribution function is obtained empirically from data. Let x_N represent N uniformly distributed random samples of the random variable x. Then, the cumulative distribution function is defined by A.1b.

$$p(x) = \frac{dP(x)}{dx} = \lim_{\delta x \to 0} \frac{P(x + \delta x) - P(x)}{\delta x} \tag{A.1a}$$

$$P(x) = \lim_{N \to \infty} \frac{count(X_N < x)}{N} \tag{A.1b}$$

Equation A.1b and A.1a effectively defines the bin used for counting samples as x to $x + \delta x$ which includes x but not $x + \delta x$. This bin width is represented by the interval $[x, x + \delta x)$

APPENDIX B

ERROR ANALYSIS FOR AREA CALCULATION WITH BINNING

At the end of 3 we note that the the ratio $V(\psi, \Delta\phi)/V\ \Delta\phi$ is an approximation to the the probability density function of ϕ at ψ, given by $P(\phi; \phi = \psi)$. We show here our reasons for that statement and also do an analysis for the order of accuracy of (3.6). Recall from 3 that $A(\psi)$ denotes the exact area of the surface with $\phi = \psi$. The ratio $V(\psi, \Delta\phi)/V$ is the ratio of the number of locations with scalar value between ψ and $\psi + \Delta\phi$ to the total number of locations, which equals the probability $p(\phi; \psi \leq \phi < \psi + \Delta\phi)$ of finding a point with scalar value in the range $[\phi, \phi + \Delta\phi)$. This equivalence of $V(\psi, \Delta\phi)/V$ with the probability of finding a position $\vec{x}$ such that $\phi(\vec{x}) \in [\psi, \psi + \Delta\phi)$ follows from $\vec{x}$ being a uniformly distributed random variable. If $P(\phi)$ is the p.d.f. of ϕ then, from the definition of a p.d.f.,

$$\frac{V(\psi, \Delta\phi)}{V} = p(\phi; \psi \leq \phi + \Delta\phi) = \int_{\psi}^{\psi+\Delta\phi} P(\phi) d\phi \tag{B.1}$$

Using the rectangle rule, the left hand side in (3.4) be written

$$\int_{\psi}^{\psi+\Delta\phi} A(\phi) d\phi = A(\psi)\Delta\phi + k_A(\Delta\phi)\Delta\phi^2 \ , \tag{B.2}$$

where $k_A(\Delta\phi)$ denotes the maximum value of $dA(\phi)d\phi$ in the domain $[\psi, \psi + \Delta\phi)$. Using (3.4), (B.1) and (B.2), we arrive at

$$\frac{A(\psi)\Delta\phi + k_A(\Delta\phi)\Delta\phi^2}{V} = \int_{\psi}^{\psi+\Delta\phi} P(\phi) d\phi \ \langle |\nabla\phi| \rangle_{V(\psi,\Delta\phi)} \ . \tag{B.3}$$

Dividing through by $\Delta\phi$ yields,

$$\frac{A(\psi)}{V} = \frac{\int_{\psi}^{\psi+\Delta\phi} P(\phi)d\phi}{\Delta\phi} \langle|\nabla\phi|\rangle_{V(\psi,\Delta\phi)} - \frac{k_A(\Delta\phi)}{V}\Delta\phi \tag{B.4}$$

which is a first order approximation.

Finally, approximating the integral of the p.d.f. with the rectangle rule results in

$$\frac{A(\psi)}{V} = \frac{P(\phi;\phi=\psi)\Delta\phi + k_p(\Delta\phi)\Delta\phi^2}{\Delta\phi} \langle|\nabla\phi|\rangle_{V(\psi,\Delta\phi)} - \frac{k_A(\Delta\phi)}{V}\Delta\phi \tag{B.5}$$

where $k_P(\Delta\phi)$ is the maximum value of $dP(\phi)d\phi$ in the domain $[\psi, \psi+\Delta\phi)$. By comparing (B.5) and (3.6) we see that, if the Monte-Carlo integration of the latter is converged then it provides a first-order approximation of $P(\phi;\phi=\psi)$.

APPENDIX C

RELATION BETWEEN AREA INTEGRAL AND VOLUME INTEGRAL EQUATION FOR ISO-SURFACE AREA

Consider a scalar field $\phi(\vec{x})$ to be a function that maps $\vec{x}$ to a ϕ. If we randomly pick a location $\vec{\xi}$ then the probability of $\phi(\vec{\xi}) = \psi$ is the same as the probability of picking a value ψ in the field. This is the basis for deriving an integral method for the area to volume ratio. Formal derivations are in Pope [47] and Yurtoglu et al. [78]. Here we provide a physical justification for the approach.

The volume in which we want the surface to volume ratio is V, the set of all possible position vectors in V is $\mathbf{X}$, and the set of all values of ϕ in V is $\boldsymbol{\phi}$. The subset of $\boldsymbol{X}$ at which $\phi = \psi$ is $\boldsymbol{X}_\psi$. Assuming that an isosurface exists with isovalue ψ then its area is A_ψ and the differential area associated with each point in $\boldsymbol{X}_\psi$ is $\hat{n}dA$ where $\hat{n} = \nabla\phi/\left|\nabla\phi\right|$. The volume between isosurfaces with $\phi = \psi$ and $\phi = \psi + d\phi$ corresponds to the set of all the position vectors between $\boldsymbol{X}_\psi$ and $\boldsymbol{X}_{\psi+d\phi}$.

By definition, an iso-surface is perpendicular to $\nabla\phi$ with normal vector $\hat{n}$; the notation is illustrated in figure Consider a position vector $\vec{x} = \vec{\xi}$ such that $\vec{\xi} \in \boldsymbol{X}_\psi$. For every $\vec{\xi}$, since the gradient $\nabla\phi$ is defined at every point, there exists a differential vector $d\vec{x} = \hat{n}dx$ such that (3.1) holds. Hence, a volume element at $\vec{\xi}$ corresponding to the volume between ensembles $\boldsymbol{X}_\psi$ and $\boldsymbol{X}_{\psi+d\phi}$ is given by $\hat{n}dA \cdot \hat{n}dx$. The volume of this region is given by integrating over $\boldsymbol{X}_\psi$ as given by (3.2). It may also be noted that the expression $\int_{\boldsymbol{X}_\psi} \hat{n}dA \cdot \hat{n}q(\vec{x})$ integrates any scalar valued function $q(\vec{x})$ over $\boldsymbol{X}_\psi$. Hence, the expression

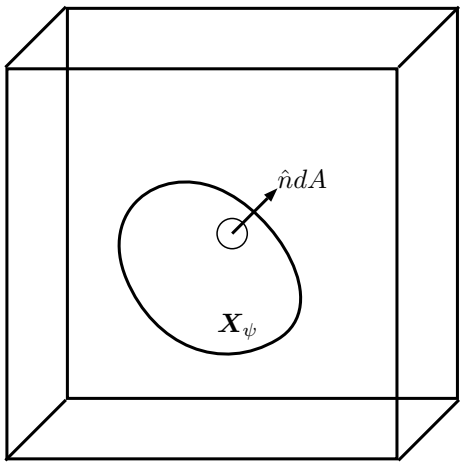

Figure C.1: Cartoon of an iso-surface in a volume to illustrate the notation.

$$\langle q(\vec{x})\rangle_{\boldsymbol{X}_\psi} \equiv \frac{\int_{\boldsymbol{X}_\psi} \hat{n} q(\vec{x}) \cdot \hat{n} dA}{A_\psi} \tag{C.1}$$

defines the notation $\langle q(\vec{x})\rangle_{\boldsymbol{X}_\psi}$ as the mean of $q(\vec{\xi})$ over $\boldsymbol{X}_\psi$. Thus, the mean of all differential coordinate vectors in $\boldsymbol{X}_\psi$ is

$$\langle dx\rangle_{\boldsymbol{X}_\psi} = \frac{\int_{\boldsymbol{X}_\psi} \hat{n} dx \cdot \hat{n} dA}{A_\psi} \ . \tag{C.2}$$

Then, from (3.2) and (C.2), the volume between the two iso-surfaces defined by $\boldsymbol{X}_\psi$ and $\boldsymbol{X}_{\psi+d\phi}$ is

$$dV = A_\psi \langle dx\rangle_{\boldsymbol{X}_\psi} \ . \tag{C.3}$$

The probability of a location $\vec{\xi}$ randomly selected from $\boldsymbol{X}$ being in the volume dV is dV/V. Let $p(\phi \in [\phi, \psi + d\phi])$ denote the probability of a randomly picked value of scalar ϕ being in between ψ and $\psi + d\phi$. Since the probability of $\vec{\xi}$ such that $\phi(\vec{\xi}) \in [\psi, \psi + d\phi]$ is the same as the probability of a randomly selected ϕ being between ψ and $\psi + d\phi$,

$$p(\phi \in [\phi, \psi + d\phi))] = \frac{dV}{V} = \frac{A_\psi \langle dx\rangle_{\boldsymbol{X}_\psi}}{V} \ . \tag{C.4}$$

It is important to note that while the left hand side of (C.4) is written in terms of a probability, the area to volume ratio is deterministic; the properties of random variables have been introduced for convenience, not because the iso-surface area is stochastic. Having introduced the probability $p(\phi \in [\phi, \psi + d\phi))]$, we observe that $p(\phi \in [\phi, \psi + d\phi)) = P(\phi; \phi = \psi)d\phi$, where $P(\phi; \phi = \psi)$ is the probility density of ϕ taking on the value ψ. With this notation, (C.4) reduces to

$$P(\phi; \phi = \psi) = \frac{A_\psi}{V} \left\langle \frac{dx}{d\phi} \right\rangle_{\boldsymbol{X}_\psi} \tag{C.5}$$

where $\langle dx/d\phi \rangle_{\boldsymbol{X}_\psi}$ is the mean of on $\boldsymbol{X}_\psi$ and has a specific value, that is, it is a number. We are interested in the mean of $d\phi/dx$ over $\boldsymbol{X}_\psi$. For simplicity, let $d\phi/dx$ at any point on $\boldsymbol{X}_\psi$ be given by Z. Let $\left\langle \frac{d\phi}{dx} \right\rangle_{\boldsymbol{X}_\psi}$ be the expected mean of Z over $\boldsymbol{X}_\psi$, and be represented by $E(Z)$.As $d\vec{x} = \hat{n}dx$, $\frac{d\phi}{dx}$ is equal to $|\nabla\phi|$. Let $dx/d\phi$ at any point on $\boldsymbol{X}_\psi$be given by Y. Thus Y is a function of Z, given as $Y = Y(Z) = 1/Z$, because the direction of dx is the same as the gradient and represents a monotonic increase in ϕ at any point $\boldsymbol{X}_\psi$. The expected means for both variables are given by (C.6)

$$E(Z) = \int_{\boldsymbol{X}_\psi} Z \; P(Z) \; dZ = \left\langle \frac{d\phi}{dx} \right\rangle_{\boldsymbol{X}_\psi} = \langle |\nabla\phi| \rangle_{\boldsymbol{X}_\psi} = \mu_1 \tag{C.6a}$$

$$E(Y) = \int_{\boldsymbol{X}_\psi} Y \; P(Z) \; dZ = \left\langle \frac{dx}{d\phi} \right\rangle_{\boldsymbol{X}_\psi} \tag{C.6b}$$

$Y(Z)$ can be expanded about the mean value of Z, i.e μ_1 using a Taylor Series expansion, given by (C.7)

$$Y(Z) = Y(\mu_1) + \frac{dY(\mu_1)}{dZ}(Z - \mu_1) + \frac{1}{2!}\frac{d^2Y(\mu_1)}{dZ^2}(Z - \mu_1)^2 + \frac{1}{3!}\frac{d^3Y(\mu_1)}{dZ^3}(Z - \mu_1)^3 + \frac{1}{4!}\frac{d^4Y(\mu_1)}{dZ^4}(Z - \mu_1)^4 + \cdots \tag{C.7}$$

Substituting (C.7) in (C.6b) and using (C.6a) and the fact that $\int_{\boldsymbol{X}_\psi} P(Z)dz = 1$we get (C.8).

$$\begin{aligned} E(Y) = \int_{\boldsymbol{X}_\psi} (\, Y(\mu_1) + \frac{dY(\mu_1)}{dZ}(Z-\mu_1) + \frac{1}{2!}\frac{d^2Y(\mu_1)}{dZ^2}(Z-\mu_1)^2 + \frac{1}{3!}\frac{d^3Y(\mu_1)}{dZ^3}(Z-\mu_1)^3 \\ + \frac{1}{4!}\frac{d^4Y(\mu_1)}{dZ^4}(Z-\mu_1)^4 + \cdots)P(Z)\, dZ \\ = \int_{\boldsymbol{X}_\psi} Y(\mu_1)P(Z)dz + \frac{dY(\mu_1)}{dZ}\int_{\boldsymbol{X}_\psi}(Z-\mu_1)P(Z)dZ + \frac{1}{2!}\frac{d^2Y(\mu_1)}{dZ^2}\int_{\boldsymbol{X}_\psi}(Z-\mu_1)^2P(Z)dZ \\ + \frac{1}{3!}\frac{d^3Y(\mu_1)}{dZ^3}\int_{\boldsymbol{X}_\psi}(Z-\mu_1)^3P(Z)dZ + \frac{1}{4!}\frac{d^4Y(\mu_1)}{dZ^4}\int_{\boldsymbol{X}_\psi}(Z-\mu_1)^4 + \cdots)P(Z)\, dZ \\ = Y(\mu_1) + \frac{dY(\mu_1)}{dZ}(0) + \frac{1}{2!}\frac{d^2Y(\mu_1)}{dZ^2}\mu_2(Z) + \frac{1}{3!}\frac{d^3Y(\mu_1)}{dZ^3}\mu_3(Z) + \frac{1}{4!}\frac{d^4Y(\mu_1)}{dZ^4}\mu_4(Z) + \cdots \end{aligned} \tag{C.8}$$

(C.8) can be simplified to (C.9) by substituting Y in (C.8).

$$E(Y) = \frac{1}{E(Z)} + \frac{\mu_2(Z)}{E(Z)^3} - \frac{\mu_3(Z)}{E(Z)^4} + \frac{\mu_4(Z)}{E(Z)^5} + \cdots \tag{C.9}$$

Referring to (C.6b) and substituting (C.9) in (C.5), we get (C.10)

$$\frac{A_\psi}{V} = P(\phi;\phi=\psi)\langle|\nabla\phi|\rangle_{\boldsymbol{X}_\psi}\left(\frac{1}{1+\frac{\mu_2(Z)}{\langle|\nabla\phi|\rangle^2_{\boldsymbol{X}_\psi}} - \frac{\mu_3(Z)}{\langle|\nabla\phi|\rangle^3_{\boldsymbol{X}_\psi}} + \frac{\mu_4(Z)}{\langle|\nabla\phi|\rangle^4_{\boldsymbol{X}_\psi}} + \cdots}\right) . \tag{C.10}$$

(C.10) rearranges to form an expression for the area to volume ratio for the isosurface $\phi = \psi$, if the terms in its denominator are negligible as compared to the leading term and gives (C.11).

$$\frac{A_\psi}{V} = P(\phi;\phi=\psi)\langle|\nabla\phi|\rangle_{\boldsymbol{X}_\psi} . \tag{C.11}$$

Equation (C.11) is the basis for integral methods for computing isosurface areas. For a more rigorous proof of (C.10), refer to Storti [67].

APPENDIX D

CALCULATING ERROR TERMS FOR AREA RATIO EQUATION

In 3, we show the equation for the exact calculation of iso-surface areas. In this chapter, we describe how integrals over sample space for the gradient are converted to integrals over the iso-surface. Let v_1 and v_2 represent a coordinate system on the iso-surface $\boldsymbol{X}_\psi$. The magnitude of the gradient of the scalar on $\boldsymbol{X}_\psi$ is given by $|\nabla\phi|$, which is also given by Z, as per notation used in §3. Z is a function of position on the surface $\boldsymbol{X}_\psi$. Thus, $Z = Z(v_1, v_2)$. The mean value of Z over $\boldsymbol{X}_\psi$ is given by $E(Z)$ (§3, (C.6a)). Let $P(v_1, v_2)$ be the joint p.d.f of Using LOTUS (Law of the unconscious statistician) or Papoulis [43, Equation 7.2],

$$E(Z(v_1, v_2)) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Z(v_1, v_2)P(v_1, v_2)dv_1dv_2 \tag{D.1}$$

The term $P(v_1, v_2)dv_1d_v2$ represents the probability of finding v_1 and v_2 in the region given by dv_1 and dv_2, which is an area dA on the surface $\boldsymbol{X}_\psi$. Therefore, equation (D.2) should hold.

$$P(v_1, v_2)dv_1dv_2 = dA/A_\psi \tag{D.2}$$

Using (D.1) and (D.2), we get (D.3)

$$E(Z(v_1, v_2)) = \frac{\int_{\boldsymbol{X}_\psi} Z(v_1, v_2)dA}{A_\psi} \tag{D.3}$$

(D.3) is also supported by the discrete version of LOTUS where $P(v_1, v_2)dv_1dv_2$ is replaced by the joint probability mass function $P(v_1, v_2)$ which is the probability of finding the exact discrete variables v_1 and v_2.

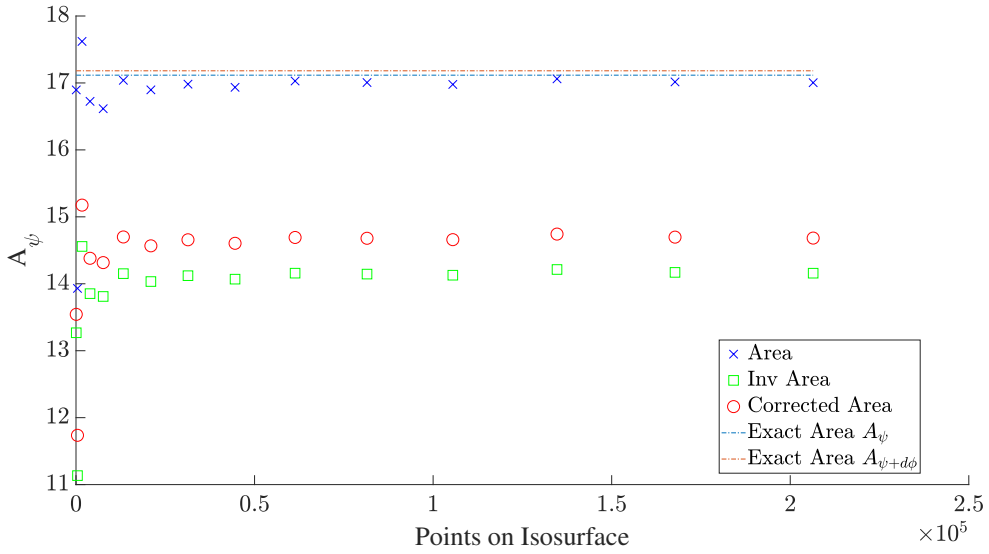

Figure D.1: Area for the test function in §3 was calculated with a bin width 0.01, i.e. between $\psi = 4$ and $\psi = 4.01$. The plot shows the area calculated with the exact ((C.10)) and simplified ((C.11)) equations given in §3

Note that the function Z in (D.1) can be replaced by any function defined over $\boldsymbol{X}_\psi$, such as $(Z - E(Z))^2$, which is the variance of the gradient. For the test function in §3, we calculated first four terms from (C.10) with increasing grid size and a bin width and plotted it with the exact area, which can be seen in figure D.1.

APPENDIX E

SCRIPTS USED FOR CHAPTERS 4 AND 5

This chapter contains scripts that were used to perform calculations for certain specialized testing. F contains the C++ code that was used for all the remaining calculations.

E.1 Figure 4.3

```
clear all;
%%Setup and plot for one iso-value
clear all;
niso=2; %SET Iso Value to Plot
bw=1e-2;
Lx = 2*pi();

Nintg = 28.3064534922624/(Lx)^3; % from marching cubes
    converged answer for 550^3 points
%%Loop
for gridsize=1:11
    Nx(gridsize) = gridsize*50; %gridsize variable for
        plotting
    nxyz=(gridsize*50 - 1); %SET Grid Size
    dx = Lx/(nxyz);
```

```
14      X = [-Lx/2:dx:Lx/2]; Y = [-Lx/2:dx:Lx/2]; Z = [-Lx/2:dx:
            Lx/2];
15      %dx = Lx/(nxyz+1);
16      %X = [-Lx/2:dx:Lx/2-dx]; Y = [-Lx/2:dx:Lx/2-dx]; Z = [-Lx
            /2:dx:Lx/2-dx];
17      A = zeros(nxyz+1,nxyz+1,nxyz+1);
18      GA = zeros(nxyz+1,nxyz+1,nxyz+1);
19      parfor i=1:nxyz+1
20          for j=1:nxyz+1
21              for k=1:nxyz+1
22                  A(i,j,k) = (cos(X(i))) + (cos(Y(j))) + (cos(Z
                        (k)));
23                  GA(i,j,k) = sqrt(sin(X(i))^2 + sin(Y(j))^2 +
                        sin(Z(k))^2);
24              end
25          end
26      end
27
28      %%Statistical Method
29      [isoindex_x,isoindex_y,isoindex_z] = ind2sub(size(A),find
            (A>=niso & A<niso+bw));
30      G = zeros(numel(isoindex_x),1);
31      for i=1:numel(isoindex_x)
32          G(i)=GA(isoindex_x(i),isoindex_y(i),isoindex_z(i));
33      end
34      gniso=mean(G);
35      pdf=numel(isoindex_x)/((nxyz+1)^3 * bw);
```

```
36      points = numel(isoindex_x);
37      Area = gniso * pdf * Lx^3;
38      StatArea(gridsize) = Area;
39      %}
40
41      %%Marching Cubes
42      [x,y,z] = meshgrid(X,Y,Z);
43      [F,V] = MarchingCubes(x,y,z,A,niso);
44      clear x y z;
45      parfor i=1:numel(F(:,1)) side1(i,:) = V(F(i,1),:) - V(F(i
            ,2),:); end
46      parfor i=1:numel(F(:,1)) side2(i,:) = V(F(i,1),:) - V(F(i
            ,3),:); end
47      fa=cross(side1,side2,2)/2;
48      parfor i=1:numel(F(:,1)) absfa(i) = sqrt(dot(fa(i,:),fa(i
            ,:))); end
49      MarchArea(gridsize) = sum(absfa);
50
51
52      [x,y,z] = meshgrid(X,Y,Z);
53      [F,V] = isosurface(x,y,z,A,niso); %The matlab function
54      clear x y z;
55      parfor i=1:numel(F(:,1)) side1(i,:) = V(F(i,1),:) - V(F(i
            ,2),:); end
56      parfor i=1:numel(F(:,1)) side2(i,:) = V(F(i,1),:) - V(F(i
            ,3),:); end
57      fa=cross(side1,side2,2)/2;
```

```
58      parfor i=1:numel(F(:,1)) absfa(i) = sqrt(dot(fa(i,:),fa(i
            ,:))); end
59      IsoArea(gridsize) = sum(absfa);
60
61      clear absfa side1 side2 fa F V  Y Z A GA dx G isoindex_x
            isoindex_y isoindex_z
62 end
63
64 figure();
65 set(groot, 'defaultAxesTickLabelInterpreter','latex', '
       defaultLegendInterpreter','latex');
66 semilogy(Nx, abs(StatArea-IsoArea(end))/IsoArea(end),"sg",'
       DisplayName','Integral Method','MarkerSize',20,'Linewidth'
       ,2);
67 hold on;
68 semilogy(Nx, abs(IsoArea-IsoArea(end))/IsoArea(end),"xb",'
       DisplayName','Marching Cubes','MarkerSize',20,'Linewidth'
       ,2);
69 xlabel('Grid size N');
70 ylabel(['Relative error in $\frac{A_\psi}{V}$'],'Interpreter'
       ,'latex');
71 legend(gca,'show','Location','best');
72 set(gca,'FontSize',32,'FontName','Times');
73 savefig("./RelativeError_MC_and_Integral.fig");
```

E.2 Figure 4.4 and 4.8

```
1 clear all;
```

```
2 niso=2; %SET Iso Value to Plot
3 bw=1e-2;
4 Lx = 2*pi();
5 %%Numerical Area
6 warning('off','all')
7 IArea = @(x,y) (1+(sin(x).^2+sin(y).^2)./(1-(niso-cos(x)-cos(
      y)).^2)).^(1/2).*((cos(x)+cos(y))>=(niso-1));
8 Nintg=2*integral2(IArea,-Lx/2,Lx/2,-Lx/2,Lx/2);
9 Nintg;
10
11 %%Loop
12 for gridsize=1:60
13     nxyz=(gridsize*10 - 1); %SET Grid Size
14     Nx(gridsize) = nxyz+1;
15     dx = Lx/nxyz;
16     X = [-Lx/2:dx:Lx/2]; Y = [-Lx/2:dx:Lx/2]; Z = [-Lx/2:dx:
          Lx/2];
17     A = zeros(nxyz+1,nxyz+1,nxyz+1);
18     GA = zeros(nxyz+1,nxyz+1,nxyz+1);
19     parfor i=1:nxyz+1
20         for j=1:nxyz+1
21             for k=1:nxyz+1
22                 A(i,j,k) = (cos(X(i))) + (cos(Y(j))) + (cos(Z
                      (k)));
23                 GA(i,j,k) = sqrt(sin(X(i))^2 + sin(Y(j))^2 +
                      sin(Z(k))^2);
24             end
```

```
25          end
26      end
27      %%PDF with fixed binwidth
28      [isoindex_x,isoindex_y,isoindex_z] = ind2sub(size(A),find
            (A>=niso & A<niso+bw));
29      pdf=numel(isoindex_x)/((nxyz+1)^3 * bw);
30      StatPdf(gridsize) = pdf;
31      StatNp(gridsize) = numel(isoindex_x);
32      %Average Gradient with fixed binwidth
33      G = zeros(numel(isoindex_x),1);
34      for i=1:numel(isoindex_x)
35          G(i)=GA(isoindex_x(i),isoindex_y(i),isoindex_z(i));
36      end
37      StatGmean(gridsize)=mean(G);
38
39      %%PDF with Freedman Diaconis
40      clear isoindex_x isoindex_y isoindex_z;
41      bw1 = 2*iqr(A(:))/(nxyz+1); %Cuberoot of number of points
42      [isoindex_x,isoindex_y,isoindex_z] = ind2sub(size(A),find
            (A>=niso & A<niso+bw1));
43      pdf = numel(isoindex_x)/((nxyz+1)^3 * bw1);
44      FdPdf(gridsize) = pdf;
45      FdNp(gridsize) = numel(isoindex_x);
46
47      clear X Y Z A GA dx G isoindex_x isoindex_y isoindex_z
            nxyz;
48  end
```

```
49
50 %%Plotting PDF for our methods(Fixed Bin Width) vs Freedman
       Diaconis
51 figure();
52 hold on;
53 plot(Nx,FdPdf,'or','DisplayName','Freedman-Diaconis rule bin
       width','MarkerSize',20);
54 plot(Nx,StatPdf,'xb','DisplayName',sprintf('Fixed bin width')
       ,'MarkerSize',20);
55 xlabel('Grid Size ($N$, Grid of $N^3$)');
56 ylabel(['$P(\phi;\phi=\psi)$']);
57 %title(['Convergence of PDF of \phi=',num2str(niso),'with two
        different binning methods'],'FontWeight','normal');
58 legend(gca,'show','Location','best');
59 set(gca,'FontSize',30,'FontName','Times');
60 hold off;
61 savefig('PDFwithmethods');
62
63 %%Plotting Np
64 figure();
65 set(groot, 'defaultAxesTickLabelInterpreter','latex', '
       defaultLegendInterpreter','latex');
66 hold on;
67 plot(Nx,FdNp,'or','DisplayName','Freedman-Diaconis rule bin
       width','MarkerSize',20);
68 plot(Nx,StatNp,'xb','DisplayName',sprintf('Fixed bin width'),
       'MarkerSize',20);
```

```
69 xlabel('Grid Size ($N$, Grid of $N^3$)');
70 ylabel('Points in bin');
71 %title(['Points in bin for \phi=',num2str(niso),'with two
       different bin width calculation methods'],'FontWeight','
       normal');
72 legend(gca,'show','Location','best');
73 set(gca,'FontSize',30,'FontName','Times');
74 savefig('PointsinBin');
75 hold off;
76
77 %Plotting PDF vs square root of number of samples, with fixed
       binwidth
78 figure();
79 set(groot, 'defaultAxesTickLabelInterpreter','latex', '
       defaultLegendInterpreter','latex');
80 loglog((Nx.^3),(abs((StatPdf-StatPdf(end))/StatPdf(end))),'xb
       ','DisplayName','Fixed bin width','MarkerSize',20);
81 model = [(Nx(1)^(3/2))./(Nx.^3).^(1/2) Nx(1)^(3/2)/(1000)
       ^(3/2)];
82 hold on;
83 loglog([Nx.^3 1000^3],model,'-b','DisplayName','$1/(N^3)
       ^{1/2}$');
84 xlabel('$Log(N^{3})$');
85 ylabel(['Relative Error in $P(\phi;\phi=\psi)$']);
86 %title(['PDF Convergence for \phi=',num2str(niso),' vs number
       of samples'],'FontWeight','normal');
87 legend(gca,'show','Location','best');
```

```
set(gca,'FontSize',30,'FontName','Times');
savefig('PDFConvergence');

%Plotting Average Gradient with Fixed binwidth, normalized by
    maximum
%gradient magnitude, i.e.
gmax = (3)^(1/2);
figure();
set(groot, 'defaultAxesTickLabelInterpreter','latex', '
    defaultLegendInterpreter','latex');
plot(Nx,StatGmean/gmax,'xb','DisplayName',['Fixed bin width '
    ],'MarkerSize',20);
xlabel('Grid Size ($N$, Grid of $N^3$)');
ylabel(['$\langle |\nabla \phi| \rangle_{\mathbf{X}_\psi}/|\
    nabla \phi|_{max}$' ]);
legend(gca,'show','Location','best');
set(gca,'FontSize',30,'FontName','Times');
savefig('GradientConvergence');
```

E.3 Figure 4.6

```
clear all;
niso=2; %SET Iso Value to Plot
bwr=linspace(0.005,0.1,20);
Lx = 2*pi();
%%Numerical Area
Nintg = 28.3064534922624; % from marching cubes converged
    answer for 550^3 points
```

```
7
8 %%Loop
9 for gridsize=1:2
10     nxyz=(gridsize*250 - 1); %SET Grid Size
11     Nx(gridsize) = nxyz+1;
12     dx = Lx/nxyz;
13     X = [-Lx/2:dx:Lx/2]; Y = [-Lx/2:dx:Lx/2]; Z = [-Lx/2:dx:
           Lx/2];
14     A = zeros(nxyz+1,nxyz+1,nxyz+1);
15     GA = zeros(nxyz+1,nxyz+1,nxyz+1);
16     parfor i=1:nxyz+1
17         for j=1:nxyz+1
18             for k=1:nxyz+1
19                 A(i,j,k) = (cos(X(i))) + (cos(Y(j))) + (cos(Z
                       (k)));
20                 GA(i,j,k) = sqrt(sin(X(i))^2 + sin(Y(j))^2 +
                       sin(Z(k))^2);
21             end
22         end
23     end
24
25     %%Statistical Method
26     %%Binwidth Loop
27     for bwi=1:numel(bwr)
28         bw = bwr(bwi);
29         [isoindex_x,isoindex_y,isoindex_z] = ind2sub(size(A),
               find(A>=niso & A<niso+bw));
```

```
30          G = zeros(numel(isoindex_x),1);
31          for i=1:numel(isoindex_x)
32              G(i)=GA(isoindex_x(i),isoindex_y(i),isoindex_z(i)
                    );
33          end
34          gniso=mean(G);
35          pdf=numel(isoindex_x)/((nxyz+1)^3 * bw);
36          Area = gniso * pdf * Lx^3;
37          StatArea(gridsize,bwi) = Area;
38          StatPdf(gridsize,bwi) = pdf;
39          StatGrad(gridsize,bwi) = gniso;
40          clear isoindex_x isoindex_y isoindex_z
41      end
42      clear X Y Z A GA dx G;
43 end
44 RelErStat = abs(StatArea-Nintg)/Nintg;
45 %%Plot Area vs binwidth
46 figure();
47 hold on;
48 phirange = 6; %Range of Scalar
49 set(groot, 'defaultAxesTickLabelInterpreter','latex', '
      defaultLegendInterpreter','latex');
50 plot(bwr/phirange,RelErStat(1,:),'or','DisplayName',sprintf('
      $%d^{3}$ points',Nx(1)),'Markersize',20,'Linewidth',2);
51 plot(bwr/phirange,RelErStat(2,:),'xb','DisplayName',sprintf('
      $%d^{3}$ points',Nx(2)),'Markersize',20,'Linewidth',2);
```

```
52 xlabel('Non-Dimensional Bin width $\Delta \phi /(\phi_{max}-\
    phi_{min})$','Interpreter','latex');
53 ylabel(['Relative error in $\frac{A_\psi}{V}$'],'Interpreter'
    ,'latex');
54 legend(gca,'show','Location','best');
55 set(gca,'FontSize',32,'FontName','Times');
56 hold off;
57 savefig('AreavsBinwidth');
```

E.4 Figure 5.6

```
1 clear all;
2 bw = [0.1 0.01 0.001];
3 niso=0;
4 gridsize = 10000; %grid size
5 intp = 1000;
6 PDF = zeros(numel(intp),numel(bw));
7 GRAD = zeros(numel(intp),numel(bw));
8 Area = zeros(numel(intp),numel(bw));
9 Points = zeros(numel(intp),numel(bw));
10 xmin=-pi();xmax=pi(); Lx = xmax - xmin;
11 ymin=-pi();ymax=pi(); Ly = ymax - ymin;
12 a=-1-pi(); b=1+pi(); %min and max values of field in above
    domain
13 A = zeros(gridsize,gridsize);
14 GA = zeros(gridsize,gridsize);
15 tic
16 for k=1:intp
```

```
17      k
18      dx = Lx/(gridsize); dy = Ly/(gridsize); %We divide by
            gridsize rather than (gridsize - 1), because we want
            the endpoint of our grid to be one spacing off from
            the domain endpoint, so that when we interpolate by
            shifting one gridpoint, we are still in the domain
19      X = [-Lx/2:dx:Lx/2-dx] + rand(1)*dx; Y = [-Ly/2:dy:Ly/2 -
            dy] + rand(1)*dy;
20      for i=1:gridsize
21          A(i,:) = cos(X(i)) + Y;
22          GA(i,:) = (1+(sin(X(i)))^2)^(1/2);
23      end
24
25      for j=1:numel(bw)
26          [isoindex_x,isoindex_y] = ind2sub(size(A),find(A>=
                niso & A<niso+bw(j)));
27          G = zeros(numel(isoindex_x),1);
28          for i=1:numel(isoindex_x)
29              G(i) = GA(isoindex_x(i),isoindex_y(i));
30          end
31          if k==1
32              Points(k,j) = numel(isoindex_x);
33              PDF(k,j) = Points(k,j)/(numel(A) * bw(j) );
34              GRAD(k,j) = sum(G)/Points(k,j);
35              Area(k,j) = PDF(k,j)*GRAD(k,j);
36          else
37              Points(k,j) = Points(k-1,j) + numel(isoindex_x);
```

```
38                PDF(k,j) = Points(k,j)/(k*numel(A) * bw(j) );
39                GRAD(k,j) = (sum(G) + GRAD(k-1,j)*Points(k-1,j))/
                      Points(k,j);
40                Area(k,j) = PDF(k,j)*GRAD(k,j);
41            end
42            clear G isoindex_x isoindex_y;
43        end
44        clear X Y dx dy; A(:) = 0; GA(:) = 0;
45 end
46 toc
47 y = @(x) (1 + (sin(x)).^2).^(1/2);
48 Nintg = integral(y,-pi(),pi()) / ((xmax-xmin)*(ymax-ymin));
49 RelEr = abs(Area-Nintg)/Nintg;
50 range = "$(2+2\pi)$";
51 % Plot
52 figure();
53 set(groot, 'defaultAxesTickLabelInterpreter','latex', '
       defaultLegendInterpreter','latex');
54 symlist = ["xb", "sg", "+r"];
55 for i=1:numel(bw)
56        loglog(Points(1:1:intp,i), RelEr(1:1:intp,i),symlist(i),'
             DisplayName',strcat(num2str(bw(i)),"/",range),'
             MarkerSize',20,'Linewidth',2);
57        hold on;
58 end
59
60 xlabel('n, number of samples');
```

```
61 ylabel(['Relative error in $\frac{A_\psi}{V}$'],'Interpreter'
     ,'latex');
62 legend(gca,'show','Location','best');
63 set(gca,'FontSize',32,'FontName','Times');
64 savefig(strcat("./
     Varying_Binwidth_random_sampling_sinewave_accumulate.fig")
     );
```

E.5 Figure 5.10 and 5.11

```
1 clear all;
2 bw = [0.0001:0.0001:0.0009  0.001:0.001:0.009  0.01:0.01:0.1];
3 niso=0.5;
4 gridsize = 10000; %grid size
5 intp = 10000;
6 PDF = zeros(numel(intp),numel(bw));
7 GRAD = zeros(numel(intp),numel(bw));
8 Area = zeros(numel(intp),numel(bw));
9 Points = zeros(numel(intp),numel(bw));
10 xmin = 0; xmax = 1; Lx = xmax - xmin;
11 ymin = 0; ymax = 1; Ly = ymax - ymin;
12 a = 0; b = 1; %min and max values of field in above domain
13 A = zeros(gridsize,gridsize);
14 GA = zeros(gridsize,gridsize);
15 tic
16 for k=1:intp
17     k
```

```
18  dx = Lx/(gridsize); dy = Ly/(gridsize); %We divide by
        gridsize rather than (gridsize - 1), because we want
        the endpoint of our grid to be one spacing off from
        the domain endpoint, so that when we interpolate by
        shifting one gridpoint, we are still in the domain
19  X = [xmin:dx:xmax-dx] + rand(1)*dx; Y = [ymin:dy:ymax -
        dy] + ((rand(1)+ rand())/2)*dy;
20  for i=1:gridsize
21      GA(i,:) = sqrt(1 + 4*X(i)^2);
22      A(i,:) = (- X(i)^2 + Y);
23  end
24
25  for j=1:numel(bw)
26      [isoindex_x,isoindex_y] = ind2sub(size(A),find(A>=(
            niso - bw(j)/2) & A<(niso + bw(j)/2) ));%Midpoint
            Kernel
27      %[isoindex_x,isoindex_y] = ind2sub(size(A),find(A>=(
            niso) & A<(niso + bw(j)) ));%Standard Kernel
28      G = zeros(numel(isoindex_x),1);
29      for i=1:numel(isoindex_x)
30          G(i) = GA(isoindex_x(i),isoindex_y(i));
31      end
32      if k==1
33          Points(k,j) = numel(isoindex_x);
34          PDF(k,j) = Points(k,j)/(numel(A) * bw(j) );
35          GRAD(k,j) = sum(G)/Points(k,j);
36          Area(k,j) = PDF(k,j)*GRAD(k,j);
```

```
37          else
38              Points(k,j) = Points(k-1,j) + numel(isoindex_x);
39              PDF(k,j) = Points(k,j)/(k*numel(A) * bw(j) );
40              GRAD(k,j) = (sum(G) + GRAD(k-1,j)*Points(k-1,j))/
                    Points(k,j);
41              Area(k,j) = PDF(k,j)*GRAD(k,j);
42          end
43          clear G isoindex_x isoindex_y;
44      end
45      clear X Y dx dy; A(:) = 0; GA(:) = 0;
46 end
47 toc
48 y = @(x) ((1/2)*x.*sqrt(1 + 4*x.^2) + (1/4)*asinh(2*x));
49 Nintg = ( y( sqrt(ymax - niso) ) - y(0) ) / ((xmax-xmin)*(
     ymax-ymin));
50 %Nintg = integral(y,0,sqrt(ymax - niso)) / ((xmax-xmin)*(ymax
     -ymin));
51 RelEr = abs(Area-Nintg)/Nintg;
52 range = "$(1)$"; %Range of scalar, to non-dimensionalize bin
     width for plot
53
54 % Optimum Binwidth Calculation
55
56 % Find optimum bin widths
57 opt_bw = zeros(intp,1);
58 samples = zeros(intp,1);
59 for i=1:intp
```

```
60      temp = numel(bw);
61      for j=1:(numel(bw)-1)
62          if (RelEr(i,temp-1)<= RelEr(i,temp))
63              temp = temp - 1;
64          else
65              break;
66          end
67          opt_bw(i) = bw(temp);
68          samples(i) = Points(i,temp);
69      end
70 end
71 [opt_bw_1,ia,ic] = unique(opt_bw,'stable');
72 tsamples = numel(A)*[1:1:intp]; tsamples = tsamples(ia);
       tsamples = tsamples';
73
74 %Plotting optimum bin width vs points
75 figure();
76 set(groot, 'defaultAxesTickLabelInterpreter','latex', '
       defaultLegendInterpreter','latex');
77
78 loglog(tsamples,opt_bw_1,'ob','DisplayName',['$\psi=$'
       num2str(niso)],'MarkerSize',20);
79 model = opt_bw_1(end)*(tsamples(end)).^(1/5)./([tsamples;
       numel(A)*intp]).^(1/5);
80 hold on;
81 loglog([tsamples; numel(A)*intp],model,'-b','DisplayName','$c
       /N^{1/5}$');
```

```
82 xlabel('Total Number of samples, N');
83 ylabel(['Optimum Bin Width']);
84 %title(['PDF Convergence for \phi=',num2str(niso),' vs number
       of samples'], 'FontWeight', 'normal');
85 legend(gca,'show','Location','best');
86 set(gca,'FontSize',30,'FontName','Times');
87 savefig('Optimumbinwidth_kernel');
88
89 % Plot Convergence
90
91 %Plot locations
92 loc = zeros(intp,numel(bw)); loc(:) = 1; loc(1,:) = 1; loc(
     intp,:) = intp; %Initialize it to plot no points except
     start and end
93 for j = 1:numel(bw)
94         a = 1; b = 2;
95         while (a<intp & b<intp)
96                 if (Points(b,j) - Points(a,j) >= 0.2*Points(a
                    ,j))
97             loc(b,j) = b;
98                         a = b;
99         end
100                 b = b + 1;
101     end
102 end
103 figure();
```

```
104 set(groot, 'defaultAxesTickLabelInterpreter','latex', '
      defaultLegendInterpreter','latex');
105 symlist = ["xb", "sg", "+r"];
106 for i=1:numel(bw)
107     loglog(Points(loc(:,i),i), RelEr(loc(:,i),i),symlist(i),'
          DisplayName',strcat(num2str(bw(i)),"/",range),'
          MarkerSize',20,'Linewidth',2);
108     hold on;
109 end
110
111 xlabel('n, number of samples');
112 ylabel(['Relative error in $\frac{A_\psi}{V}, \psi=$' num2str
      (niso)],'Interpreter','latex');
113 legend(gca,'show','Location','best');
114 set(gca,'FontSize',32,'FontName','Times');
115 savefig(strcat("./
      Varying_Binwidth_random_sampling_parabola_accumulate.fig")
      );
```

APPENDIX F

ISO-SURFACE CALCULATION CLASS - KEDAR.C

This section contains the isosurface area calculation class developed in order to integrate with the spectral code.

```
1  /* -*-c++-*- */
2
3  /*
       ************************************************************************
4    * REVISION/MODIFICATION HISTORY:
5    *
6    * $Log: kedar.C,v $
7    * Revision 1.2  2019/06/04 15:38:13  kshete
8    * Updated Thesis
9    *
10   * Revision 1.1  2019/04/23 18:11:37  kshete
11   * *** empty log message ***
12   *
13   *
14   *
15   ************************************************************************
         */
16
```

```
17 #include "bpe.h"
18 #include "densityfield.h"
19 #include "fieldops.h"
20 #include "vfield.h"
21 #include <numeric>
22 #include <algorithm>
23 #include <stdlib.h>
24 #include <iostream>
25 #include <fstream>
26 #include <sys/time.h>
27 #include <random>
28
29 extern intptr_t binSearch(Float *f, intptr_t n, Float a);
30
31 double* fpoint;
32 //Keys sorting function Added by Kedar Prashant
33 int compareCoupledSort(const void* a, const void* b, void*
      arg)
34 {    //compare function for coupledsort
35     double* tosortfield = (double*)arg;
36     const intptr_t ia = *(const intptr_t*)a;
37     const intptr_t ib = *(const intptr_t*)b;
38     double aa=tosortfield[ia];
39     double bb=tosortfield[ib];
40     int retval=0;
41     if (aa<bb)
42         retval=-1;
```

```
43     else if (aa==bb)
44         retval=0;
45     else
46         retval=1;
47     return retval;
48     //retval = (aa<bb)*-1 + (aa>bb)*1;*/
49     /*double c1 = bb - aa;
50       double c2 = aa - bb;
51       unsigned long* l1 = (unsigned long*)(&c1);
52       unsigned long* l2 = (unsigned long*)(&c2);
53       long s1=*l1>>63;
54       long s2=*l2>>63;
55       return (int)(s1-s2);*/
56 }
57
58 bool fieldcompareCoupledSort(const long long int a, const
      long long int b)
59 {    //global compare function for field coupledsort
60     double aa=fpoint[a];
61     double bb=fpoint[b];
62     return aa<bb;
63 }
64
65
66 void coupledSort(double* Array1, double* Array2, long long
      int Arsize)
```

```
67 {    //sort Array1 in conjuction with Array1 using
       compareCoupledSort compare function
68     long long int* tosortindex;
69     tosortindex=(long long int*)malloc(Arsize*sizeof(long long
           int ));
70 #pragma omp simd
71     for (long long int i=0; i<Arsize; i++)
72     {
73         tosortindex[i]=i;
74     }
75
76     //qsort_r(tosortindex,Arsize,sizeof(intptr_t),
           compareCoupledSort,Array1); //Sort using C qsort
77     fpoint = Array1;//Assigning Global Pointer -- Beware!!
78     std::sort(tosortindex, &(tosortindex[Arsize]),
           fieldcompareCoupledSort); //C++ sort using global field
            pointer
79
80     long long int* todestindex;
81     todestindex=(long long int*)malloc(Arsize*sizeof(long long
           int ));
82 #pragma omp simd
83     for (long long int i=0; i<Arsize; i++)
84     {
85         todestindex[tosortindex[i]]=i;
86     }
87     free(tosortindex);
```

```
88      double temp1;
89      double temp2;
90      int tempindex;
91      for (long long int i=0; i<Arsize; i++)
92      {
93          while (todestindex[i]!=i)
94          {
95              temp1=Array1[todestindex[i]];
96              temp2=Array2[todestindex[i]];
97              tempindex=todestindex[todestindex[i]]; //store
                    target field and index
98              Array1[todestindex[i]]=Array1[i];
99              Array2[todestindex[i]]=Array2[i];
100             todestindex[todestindex[i]]=todestindex[i]; //set
                    target values and index
101             Array1[i]=temp1;
102             Array2[i]=temp2;
103             todestindex[i]=tempindex; //set current values to
                    stored values
104         }
105     }
106     free(todestindex);
107 }
108
109 long long int kSearch(double *f, long long int n, double a)
110 {    //Kedar's Search
111     /*
```

```
112       long long int ind;
113       ind=0;
114       while ((f[ind]+1e-15)<a && ind < n)
115       {
116       ind++;
117       }
118       //printf("\nind %d n %d Process %d",ind,n,_my_pe());
119       return (ind-1);
120     */
121
122     // binSearch is in sortField
123     return binSearch(f, n, a);
124 }
125
126 class Kedar : public S3Field
127 // inheret everyting from a 3D Fourier-transformable field
128 {
129 public:
130     ofstream outlog;
131     ofstream outlogcu;
132     S3Field F; // Fourier space version of field
133     S3Field G; // gradient field
134
135     double p2,p1,p0;
136     double Area=0;
137     double GAvg=0;
138     double Pdf=0;
```

```
139     double binwidth_loc;
140     double bw_glo_max;
141     double gsum_loc;
142     double temp_binwidth;
143     double temp_gsum;
144     int boxCount;
145     intptr_t np_loc;
146     intptr_t temp_np;
147     intptr_t tnp_loc;
148     long long int Np=0;
149     long long int Tnp=0;
150
151     std::default_random_engine generator;
152
153     Kedar() //Constructor for class Kedar. This is called
            automatically when object of class Kedar is created.
154         {
155             // initialize random number generator
156             generator.seed(_Nx*_Ny*_Nz);
157         }
158
159     int doIsoValue(double isoVal, double thickness)
160         {
161             /* find all the points between isoVal and isoVal+
                    thickness.
162                 Update:
163                 gsum_loc
```

```
164             binwidth_loc
165             np_loc
166
167             return 0 if no points found, 1 otherwise
168         */
169
170         // get lower and upper locations in sorted field
171         intptr_t a_l = kSearch(f,Nx*Ny*Nz,isoVal)+1;
172         intptr_t a_h = kSearch(f,Nx*Ny*Nz,isoVal+thickness);
173
174         // moving lower index to value above/equal to lower
                limit
175         if (f[a_l] < isoVal)
176             a_l = a_l + 1;
177
178         if (a_h>=a_l) // if there are points then process
                them
179         {
180             np_loc = a_h-a_l+1;
181             binwidth_loc = f[a_h]-f[a_l];
182             gsum_loc=0;
183             double c;
184             c=0;
185             for (intptr_t i=a_l; i<=a_h; i++)
186             {   //Local kahan sum of gradient
187                 double t,y;
188                 y=G.f[i]+c;
```

```
189                 t=gsum_loc+y;
190                 c=(t-gsum_loc)-y;
191                 gsum_loc=t;
192             }
193             temp_np = temp_np + np_loc;
194             temp_gsum = temp_gsum + gsum_loc;
195             if (binwidth_loc > temp_binwidth)
196                 temp_binwidth = binwidth_loc;
197         }
198         return (a_h>a_l);
199     }
200
201 void doPhaseShift(int shift)
202     {
203         // shift is the number of previous phase shifts
204         // Phase shift field using random phase angles
205         intptr_t i,j,k, ic;
206         double pflip = 1;
207         std::uniform_real_distribution<double> distribution
                (0,1);
208
209         if(shift%2==0) {
210             pflip = -1;
211         }
212
213         //Uniform Random Phase shift
214         p2 = distribution(generator) * dz * pflip;
```

```
215             p1 = distribution(generator) * dy * pflip;
216             p0 = distribution(generator) * dx * pflip;
217
218             if(shift==0)
219             {
220                 p0=0;
221                 p1=0;
222                 p2=0;
223             }
224             double ay[yh-yl+1], by[yh-yl+1];
225 #pragma omp simd
226             for (j=yl; j<=yh; j++)
227             {
228                 ay[j-yl] = cos(k1[j]*p1);
229                 by[j-yl] = sin(k1[j]*p1);
230             }
231
232             double ax[nkx], bx[nkx];
233 #pragma omp simd
234             for (i=0; i<nkx; i++)
235             {
236                 ax[i] = cos(k0[i]*p0);
237                 bx[i] = sin(k0[i]*p0);
238             }
239 #pragma omp parallel for private(i,j,k,ic)
240             for (k=0; k < Nz; k++)//Phase shifts in z,y and x
                    direction
```

```
241         {
242             double az, bz;
243             az = cos(k2[k]*p2);
244             bz = sin(k2[k]*p2);
245             for (j=yl; j<=yh; j++)
246             {
247 #pragma omp simd
248                 for (i=0; i<(nkx); i++)
249                 {
250                     ic=indexc(i,j,k); // index into complex
                            field
251
252                     register double tmp;
253                     tmp = c[ic].r;
254                     c[ic].r = az * c[ic].r - bz * c[ic].i;
255                     c[ic].i = bz * tmp + az * c[ic].i;
256
257                     tmp = c[ic].r;
258                     c[ic].r = ay[j-yl] * c[ic].r - by[j-yl] * c
                            [ic].i;
259                     c[ic].i = by[j-yl] * tmp + ay[j-yl] * c[ic
                            ].i;
260
261                     tmp = c[ic].r;
262                     c[ic].r = ax[i] * c[ic].r - bx[i] * c[ic].i
                            ;
263                     c[ic].i = bx[i] * tmp + ax[i] * c[ic].i;
```

```
264                 }
265             }
266         }
267     }
268
269     inline Float getMeanScalar(Float k)
270     {
271         Float DZ=(_Lz/(Float)(_Nz));
272         Float z=(k-_Nz/2)*DZ;
273         return z*_scalarGradient;
274     }
275
276
277     void addMeanScalar(double p2)
278     {
279         // p2 is the vertical phase shift
280         intptr_t i,j,k, ir;
281         for (k=0; k < Nz; k++)
282             for (j=yl; j<=yh; j++)
283 #pragma omp simd
284                 for (i=0; i<(Nx); i++)
285                 {
286                     ir=index(i,j,k); // index into real field
287                     f[ir] = f[ir] + getMeanScalar(k+p2);
288                 }
289     }
290
```

```
291  void setGradient(S3Field& G)
292      {
293          S3Field Gx, Gy, Gz;
294          long i,j,k,ir;
295          grad(Gx,Gy,Gz);
296          Gx.fftb3();
297          Gy.fftb3();
298          Gz.fftb3();
299          for (ir=0; ir<fFieldSize; ir++)
300              G.f[ir]=sqrt(Gx.f[ir]*Gx.f[ir] + Gy.f[ir]*Gy.f[ir
                     ] +
301                                  Gz.f[ir]*Gz.f[ir]
302                                  + _scalarGradient*_scalarGradient);
303          G.domain=Real;
304      }
305
306  void run_planar(int argc, char* argv[], double iso1,
         double iso2,
307                  int n_iso, double db, int flag, int intp)
308      {
309          info("Attempting to open ", argv[1]);
310          open(argv[1]);
311          Field::read(); // read in but don't fft
312
313          S3Field F;  // store field in Fourier space for
                 reuse
314          fftf3();
```

```
315         copy(F);
316
317         // compute mean gradient for output later
318         setGradient(G);
319         G.setMean();
320         G.setMinMax();
321         info("mean gradient ", G.mean);
322         info("gradient min/max ", G.rmin, G.rmax);
323
324         // db is the dimensionless binwidth.
325         // Compute the dimensional bin width here
326         double dimensionalBinWidth=db*dx*G.mean;
327
328         // add planar means to field to find min/max
329         F.copy(*this);
330         fftb3();
331         setMinMax();
332         info("min/max without mean: ", rmin, rmax);
333         addMeanScalar(0);
334         setMinMax();
335         info("min/max with mean: ", rmin, rmax);
336
337         iso1=rmin+(rmax-rmin)/n_iso;
338         iso2=rmax-(rmax-rmin)/n_iso;
339         info("dimensional Binwidth: ",db);
340         info("dimensional BinWidth: ", dimensionalBinWidth);
341         info("Isovalues: ",iso1," and ",iso2);
```

```
342         info("Number of Isovalues: ", n_iso);
343         info("Number of interpolations: ", intp);
344
345         double niso; // variable for current isovalue in
                loop
346         double isorange[n_iso];
347
348         if (n_iso==1)
349             isorange[0] = iso1;
350         else
351             for (int i=0; i<n_iso; i++)
352                 isorange[i] = iso1 + i*(iso2-iso1)/(n_iso-1);
353
354         double GAVG[intp][n_iso], PDF[intp][n_iso], BW[intp
                ][n_iso],
355            MAXBW[intp][n_iso], AREA[intp][n_iso];
356         long long int NP[intp][n_iso], CNP[intp][n_iso], TNP
                [intp][n_iso],
357            CTNP[intp][n_iso];
358
359         //Output log file open
360         if (_my_pe()==0)
361         {
362             char fname[100];
363             snprintf(fname,100,"%s_OutputLog.txt",argv[1]);
364
365             //for iteration values on shifts
```

```
366         outlog.open(fname, std::ofstream::out | std::
                ofstream::trunc);
367         outlog<<"\nIntp\t zshift\t Iso\t Tnp\t Np\t Pdf\t
                 AvgGrad\t Area\t p0\t p1\t p2"<<endl;
368
369         snprintf(fname,100,"%s_OutputLog_Cumulative.txt",
                argv[1]);
370
371         //for cumulative values over phase shifts
372         outlogcu.open(fname, std::ofstream::out | std::
                ofstream::trunc);
373         outlogcu<<"\nIntp\t Iso\t CNP\t CTNP\t MBW\t PDF\
                t AVG\t AREA"
374                 <<endl;
375
376         outlog << "%% scalar min/max = " << rmin << " / "
                 << rmax << endl;
377         outlog << "%% mean gradient = " << G.mean << endl
                ;
378         outlogcu << "%% scalar min/max = " << rmin << " /
                 " << rmax
379                  <<endl;
380         outlogcu << "%% mean gradient = " << G.mean <<
                endl;
381         outlogcu << "%% dimensional BinWidth: " <<
                dimensionalBinWidth << endl;
382         outlog.close();
```

```
383             outlogcu.close();
384         }
385
386         time_t nSeconds;
387         //Interpolation Loop Starts
388
389         for (int shift=0; shift<intp; shift++)
390         {
391             nSeconds=time(0);
392             if (_my_pe()==0)
393                 fprintf(stderr, "%d  shift %s", shift,
394                         asctime(localtime(&nSeconds)));
395
396             F.copy(*this); // Copy stored field to this (
                    fourier space)
397             doPhaseShift(shift);
398             setGradient(G);
399             fftb3();
400             addMeanScalar(p2);
401             pack();
402             G.pack();
403             tnp_loc=Nx*Ny*Nz;
404             coupledSort(f,G.f,tnp_loc);
405
406
407             //Isovalue loop starts
408             for (int isonum = 0; isonum < n_iso; isonum++)
```

```
409             {
410                 niso = isorange[isonum]; //Do the isonum
                        isovalue
411
412                 np_loc=0; //remains zero if no points found
413                 binwidth_loc=0;
414                 gsum_loc=0;
415                 temp_gsum=0;
416                 temp_binwidth=0;
417                 temp_np=0;
418                 boxCount=0;
419
420                 // look in 3 boxes down and 3 boxes up
421                 // boxCount keeps track of the number of boxes
                        having
422                 // the scalar value.
423                 // Note that instead of shifting the box, we
                        shift the
424                 // scalar value.
425                 boxCount+=doIsoValue(niso-3*(_Lz*
                        _scalarGradient), dimensionalBinWidth);
426                 boxCount+=doIsoValue(niso-2*(_Lz*
                        _scalarGradient), dimensionalBinWidth);
427                 boxCount+=doIsoValue(niso-1*(_Lz*
                        _scalarGradient), dimensionalBinWidth);
428                 boxCount+=doIsoValue(niso-0*(_Lz*
                        _scalarGradient), dimensionalBinWidth);
```

```
429             boxCount+=doIsoValue(niso+1*(_Lz*
                    _scalarGradient), dimensionalBinWidth);
430             boxCount+=doIsoValue(niso+2*(_Lz*
                    _scalarGradient), dimensionalBinWidth);
431             boxCount+=doIsoValue(niso+3*(_Lz*
                    _scalarGradient), dimensionalBinWidth);
432
433             int world_size;
434             MPI_Comm_size(MPLCOMM_WORLD, &world_size);
435             MPI_Barrier(MPLCOMM_WORLD);
436             //printf("\nProcess %d np_loc %d",_my_pe(),
                    np_loc);
437
438             //Receive Buffer for number of points
439             intptr_t np_glo[world_size];
440             std::fill_n(np_glo,world_size,0);
441
442             //Create receive position array
443             int rdispls[world_size];
444             for (int i=0; i<world_size; i++) {
445                 rdispls[i]=i;
446             }
447             //Number of elements send array
448             int sendcounts[world_size];
449             std::fill_n(sendcounts,world_size,1);
450
451             //Element Send displacement array
```

```
452             int sdispls[world_size];
453             std::fill_n(sdispls,world_size,0);
454
455             //Number of Elements receive array
456             int recvcounts[world_size];
457             std::fill_n(recvcounts,world_size,1);
458
459             //receive buffer for maximum boxCount
460             int boxCount_glo[world_size];
461             std::fill_n(boxCount_glo,world_size,0);
462             MPI_Alltoallv(&boxCount,sendcounts,sdispls,
                    MPI_INT,
463                                 boxCount_glo,recvcounts,rdispls,
                                        MPI_INT,
464                                 MPI_COMM_WORLD);
465             int boxCount_max = boxCount_glo[0];
466             for (int i=0; i<world_size ; i++)
467             {    //get maximum boxCount
468                 if (boxCount_max<boxCount_glo[i])
469                 {
470                     boxCount_max=boxCount_glo[i];
471                 }
472             }
473             if(boxCount_max==0) {
474                 boxCount_max=1;    //We search atleast in 1
                        box
475             }
```

```
476
477             //ISOSURFACE AREA CALCULATION, LOCAL
478             binwidth_loc = temp_binwidth;
479             np_loc = temp_np;
480             gsum_loc = temp_gsum;
481             //calculation after forward and backward z
                    check with
482             //maximum boxCount
483             //tnp_loc = Nx*Ny*Nz * boxCount_max;
484             tnp_loc = Nx*Ny*Nz;
485
486             MPI_Alltoallv(&np_loc, sendcounts, sdispls,
                    MPI_LONG_LONG_INT,
487                           np_glo, recvcounts, rdispls,
                                MPI_LONG_LONG_INT,
488                           MPI_COMM_WORLD);
489
490             //Receive Buffer for gradient sum
491             double gsum_glo[world_size];
492             MPI_Alltoallv(&gsum_loc, sendcounts, sdispls,
                    MPI_DOUBLE, gsum_glo,
493                           recvcounts, rdispls, MPI_DOUBLE,
                                MPI_COMM_WORLD);
494
495             //Receive Buffer for total points
496             intptr_t tnp_glo[world_size];
```

```
497             MPI_Alltoallv(&tnp_loc,sendcounts,sdispls,
                    MPI_LONG_LONG_INT,
498                             tnp_glo,recvcounts,rdispls,
                                MPI_LONG_LONG_INT,
499                             MPI_COMM_WORLD);
500
501             //Receive Buffer for bin widths
502             double binwidth_glo[world_size];
503             MPI_Alltoallv(&binwidth_loc,sendcounts,sdispls
                    ,MPI_DOUBLE,
504                             binwidth_glo,recvcounts,rdispls,
                                MPI_DOUBLE,
505                             MPI_COMM_WORLD);
506
507             if (_my_pe()==0)
508             {   /* Calculating isosurface statistics after
                    collecting data
509                    from all processes */
510                 Area=0;
511                 GAvg=0;
512                 Np=0;
513                 Pdf=0;
514                 Tnp=0;
515                 //Add up iso-surface statistics
516                 for (int i=0; i<world_size; i++)
517                 {
```

```
518                 GAvg=GAvg+gsum_glo[i]; //Gradient Sum on
                        IsoSurface
519                 Np=Np+np_glo[i]; //Points on IsoSurface
520                 Tnp=Tnp+tnp_glo[i]; //Total Number of
                        Points, just to check even though we
                        know them
521             }
522             if(Np==0) {
523                 GAvg=0;
524             }
525             else {
526                 GAvg=GAvg/double(Np);
527             }
528
529             //Get Binwidth Maximum
530             bw_glo_max=binwidth_glo[0];
531             for (int i=0; i<world_size ; i++)
532             {
533                 if (bw_glo_max<binwidth_glo[i])
534                 {
535                     bw_glo_max=binwidth_glo[i];
536                 }
537             }
538             //Pdf=double(Np)/double(Tnp); Pdf=Pdf/
                    bw_glo_max; //Fixed Binwidth with
                    updated runtime binwidth
539             Pdf=double(Np)/double(Tnp);
```

```
540                 Pdf=Pdf/dimensionalBinWidth; //Fixed
                        Binwidth approach
541                 Area=GAvg*Pdf;
542
543                 /*Reporting statistics for current shift*/
544                 //cout<<"\nIntp\t zshift\t Iso\t Tnp\t Np\t
                        Pdf\t AvgGrad\t Area";
545                 char fname[100];
546
547                 //Calculation for global variables --
548                 NP[shift][isonum]=Np;
549                 TNP[shift][isonum]=Tnp;
550                 CNP[shift][isonum]=0;
551                 for(int i=0; i<=shift; i++) {
552                     CNP[shift][isonum]+=NP[i][isonum];
553                 }
554                 CTNP[shift][isonum]=0;
555                 for(int i=0; i<=shift; i++) {
556                     CTNP[shift][isonum]+=TNP[i][isonum];
557                 }
558                 BW[shift][isonum]=bw_glo_max;
559                 MAXBW[shift][isonum]=BW[0][isonum];
560                 for(int i=0; i<=shift; i++)
561                 {    //Cumulative maximum bin size
562                     if(BW[i][isonum]>MAXBW[shift][isonum])
563                     {
564                         MAXBW[shift][isonum]=BW[i][isonum];
```

```
565                     }
566                 }
567                 //PDF[shift][isonum]=double(CNP[shift][
                        isonum])/double(CTNP[shift][isonum]);
                        PDF[shift][isonum]=PDF[shift][isonum]/
                        MAXBW[shift][isonum];
568                 PDF[shift][isonum]=double(CNP[shift][isonum
                        ])/double(CTNP[shift][isonum]);
569                 PDF[shift][isonum]=PDF[shift][isonum]/
                        dimensionalBinWidth;
570                 if(shift==0)
571                 {
572                     GAVG[shift][isonum]=GAvg;
573                 }
574                 else
575                 {
576                     if(CNP[shift][isonum]==0) {
577                         GAVG[shift][isonum]=GAVG[shift-1][
                                isonum];
578                     }
579                     else {
580                         GAVG[shift][isonum] = (GAVG[shift-1][
                                isonum]*double(CNP[shift-1][isonum
                                ]) + GAvg*double(Np))/CNP[shift][
                                isonum];
581                     }
582                 }
```

```
583                     AREA[shift][isonum]=GAVG[shift][isonum]*PDF
                            [shift][isonum];
584
585                     /*Reporting cumulative statistics for
                            current shift*/
586                     //outlogcu<<"Intp\t Iso\t CNP\t CTNP\t MBW\
                            t PDF\t AVG\t AREA"<<endl;
587                 }
588
589                 // write output
590 // write at most 100 interpolations
591                 if ((shift%max(intp/100,1)==0) || (shift==(
                        intp-1)))
592                     if (_my_pe()==0)
593                     {
594                         char fname[100];
595                         snprintf(fname,100,"%s_OutputLog.txt",
                                argv[1]);
596                         ofstream outlog;
597                         outlog.open(fname, std::ofstream::out |
598                                     std::ofstream::app);
599                         outlog<<setw(15)<<left<<shift
600                               <<setw(15)<<left<<boxCount_max
601                               <<setw(15)<<left<<niso
602                               <<setw(15)<<left<<Tnp
603                               <<setw(15)<<left<<Np
604                               <<setw(15)<<left<<Pdf
```

```
605                 <<setw(15)<<left<<GAvg
606                 <<setw(15)<<left<<Area
607                 <<setw(15)<<left<<p0
608                 <<setw(15)<<left<<p1
609                 <<setw(15)<<left<<p2
610                 <<"\n";
611             outlog.close();
612
613             snprintf(fname,100,"%
                    s_OutputLog_Cumulative.txt",argv[1]);
614             ofstream outlogcu;
615             outlogcu.open(fname, std::ofstream::out
                    |
616                           std::ofstream::app);
617             outlogcu<<setw(15)<<left<<shift
618                     <<setw(15)<<left<<niso
619                     <<setw(15)<<left<<CNP[shift][
                        isonum]
620                     <<setw(15)<<left<<double(CTNP[
                        shift][isonum])
621                     <<setw(15)<<left<<MAXBW[shift][
                        isonum]
622                     <<setw(15)<<left<<PDF[shift][
                        isonum]
623                     <<setw(15)<<left<<GAVG[shift][
                        isonum]
```

```
624                     <<setw(15)<<left<<AREA[shift][
                            isonum]
625                     <<"\n";
626
627             outlogcu.close();
628
629             if (isonum==n_iso-1)
630             {
631                 FILE *fp;
632                 snprintf(fname,100,"%s_Area_intp_%d.
                        bin",
633                                 argv[1],shift);
634                 fp=fopen(fname,"w");
635                 fwrite(AREA,sizeof(AREA[0][0]),(shift
                        +1)*n_iso,fp);
636                 fclose(fp);
637
638                 snprintf(fname,100,"%s_Binwidth_intp_
                        %d.bin",
639                                 argv[1],shift);
640                 fp=fopen(fname,"w");
641                 fwrite(MAXBW,sizeof(MAXBW[0][0]),(
                        shift+1)*n_iso,fp);
642                 fclose(fp);
643
644                 snprintf(fname,100,"%s_Pdf_intp_%d.
                        bin",
```

```
645                                     argv[1], shift);
646                             fp=fopen(fname,"w");
647                             fwrite(PDF, sizeof(PDF[0][0]),(shift
                                    +1)*n_iso, fp);
648                             fclose(fp);
649
650                             snprintf(fname,100,"%s_Grad_intp_%d.
                                    bin",
651                                     argv[1], shift);
652                             fp=fopen(fname,"w");
653                             fwrite(GAVG, sizeof(GAVG[0][0]),(shift
                                    +1)*n_iso, fp);
654                             fclose(fp);
655
656                             snprintf(fname,100,"%s_Points_intp_%d
                                    .bin",
657                                     argv[1], shift);
658                             fp=fopen(fname,"w");
659                             fwrite(CNP, sizeof(CNP[0][0]),(shift
                                    +1)*n_iso, fp);
660                             fclose(fp);
661
662                     }
663                 }
664         } // end isonum loop
665
666         unpack();
```

```
667                 G.unpack();
668             }//end pf phase shift loop
669         }//end of run_planar function
670 };//end of class Kedar
671
672 int main(int argc, char* argv[])
673 {
674     if ((argc != 2) && (_my_pe()==0))
675         fatalError("Usage: kedar <field>\n");
676     pComStart(argc, argv);  // start MPI
677
678     time_t nSeconds;
679     nSeconds=time(0);
680     if (_my_pe()==0)
681         fprintf(stderr, "starting %s", asctime(localtime(&
              nSeconds)));
682
683     int flag=1; //If it's a case with linear planar mean
          gradient, then set 1. Else set 0.
684     double iso1=0.5, iso2=0.5; //define iso value
685     double db=0.00002; //define binwidth
686     int n_iso=1;
687     int intp=10;
688     ifstream isoinput;
689     isoinput.open("isoinput.txt");
690     if (!isoinput)
691         fatalError("\nisoinput.txt was not read properly");
```

```
692     else
693         info("\nisoinput.txt found and read:");
694
695     while (!isoinput.eof())
696     {
697         string header;
698         isoinput>>header;
699         if(header=="binwidth")
700             isoinput>>db;
701         if(header=="flag")
702             isoinput>>flag;
703         if(header=="iso1")
704             isoinput>>iso1;
705         if(header=="iso2")
706             isoinput>>iso2;
707         if(header=="n_iso")
708             isoinput>>n_iso;
709         if(header=="interpolations")
710             isoinput>>intp;
711     }
712
713 //Flag ==1 means planar mean gradient case
714     info("\nPlanar Mean Case Flag specified is: ",flag);
715     input(); // read globals and input to get simulation
            parameters
716     Kedar m;
717     if (flag==1)
```

```
718     {
719         m.run_planar(argc, argv, iso1, iso2, n_iso, db, flag,
                intp);
720     }
721     else {
722         info("non-planar case not implemented");
723         //double niso=0.5;
724         //m.run(argc, argv, niso, db, flag);
725     }
726     pComStop(); // stop MPI
727 }
```

APPENDIX G

SPECTRAL CODE DOCUMENTATION

This chapter contains a short description of the spectral code written by Professor de Bruyn Kops. The spectral code was indexed with the source code indexing tool cscope, available here `http://cscope.sourceforge.net/`. Below is a list of important variables.

- S3Field: Three dimensional field class. The isosurface calculation class Kedar inherits from this.
- dz, dy and dx: Grid spacing in z, y and x directions
- _Nx, _Ny, _Nz: Global number of points in the x, y and z directions.
- Nx, Ny and Nz: Number of points in each direction on local process
- fftf3(): Class function in S3Field that converts field to Fourier space by performing a FFT in all three directions
- fftb3(): Class function in S3Field that converts field to real space by performing an inverse FFT in all three directions
- k0, k1 and k2: Wave numbers in x, y and z directions. Each array is the size of field, and is defined globally. On individual processes, the wave numbers can be accessed by using the global indices of the points on that process. Data is sliced in the y direction when distributed across MPI.

...

APPENDIX H

CFD HUMOUR

Various CFD'isms learned at the lab. This chapter is intended as an instructive and humorous end to the thesis.

- All CFD is wrong. Some of it is useful.
- It is better to have the right answer to the wrong problem rather than having the wrong answer to the right problem. Because the right answer is worth something, and you can go to the right problem from there.
- Either we get the right answer, or we show that it cannot be done. One way or another, we always reach a conclusion.
- Stacking up the best in show for different numerical methods into one big method does not make the best method.
- We spend a hundred hours designing an algorithm, another hundred hours writing the code and one hour designing the test case. Then we wonder why the test case is wrong.
- If you cannot write it, it's probably wrong.
- If you need to explain something over and over again, it is probably wrong.
- Contour plots are evil and must be handled with utmost care.
- A good test case is zero. If the code can take zero input and predict zero output, that is a good start. Not all codes can do that.

BIBLIOGRAPHY

[1] Almalkie, S. and de Bruyn Kops, S. M. (2012a). Energy dissipation rate surrogates in incompressible Navier-Stokes turbulence. *J. Fluid Mech.*, 697:204–236.

[2] Almalkie, S. and de Bruyn Kops, S. M. (2012b). Kinetic energy dynamics in forced, homogeneous, and axisymmetric stably stratified turbulence. *J. Turbul.*, 13(29):1–29.

[3] Bohr, M. (2007). A 30 year retrospective on dennard's mosfet scaling paper. *IEEE Solid-State Circuits Society Newsletter*, 12(1):11–13.

[4] Boyd, J. P. (2001). *Chebyshev and Fourier Spectral Methods*. Dover.

[5] Bray, K. (2016). Laminar flamelets in turbulent combustion modeling. *Combustion Science and Technology*, 188(9):1372–1375.

[6] Bray, K. N. and Swaminathan, N. (2006). Scalar dissipation and flame surface density in premixed turbulent combustion. *Comptes Rendus - Mecanique*, 334(8-9):466–473.

[7] Catrakis, H. J., Aguirre, R. C., and Ruiz-Plancarte, J. (2002). Area-volume properties of fluid interfaces in turbulence: Scale-local self-similarity and cumulative scale dependence. *Journal of Fluid Mechanics*, 462:245–254.

[8] Chaudhuri, S., Kolla, H., Dave, H. L., Hawkes, E. R., Chen, J. H., and Law, C. K. (2017). Flame thickness and conditional scalar dissipation rate in a premixed temporal turbulent reacting jet. *Combustion and Flame*.

[9] Corrsin, S. (1951). On the spectrum of isotropic temperature fluctuations in an isotropic turbulence. *J. Appl. Phys.*, 22:469–472.

[10] Corrsin, S. and Kistler, A. L. (1955). Free-stream boundaries of turbulent flows. *NACA Report*, 1224:1033–1064.

[11] de Bruyn Kops, S. M. (2015). Classical turbulence scaling and intermittency in stably stratified Boussinesq turbulence. *J. Fluid Mech.*, 775:436–463.

[12] de Bruyn Kops, S. M. and Riley, J. J. (1998). Scalar transport characteristics of the linear-eddy model. *Combust. Flame*, 112(1/2):253–260.

[13] de Bruyn Kops, S. M. and Riley, J. J. (2001). Mixing models for large-eddy simulation of non-premixed turbulent combustion. *J. Fluids. Eng.*, 123(2):341–346.

[14] de Bruyn Kops, S. M. and Riley, J. J. (2019). The effects of stable stratification on the decay of initially isotropic homogeneous turbulence. *J. Fluid Mech.*, 860:787–821.

[15] Delichatsios, M. A. (1987). Air entrainment into buoyant jet flames and pool fires. *Combustion and Flame*, 70(1):33–46.

[16] Dennard, R. H., Gaensslen, F. H., Rideout, V. L., Bassous, E., and LeBlanc, A. R. (1974). Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268.

[17] Dimov, I. T., Penzov, A. A., and Stoilova, S. S. (2007). Parallel monte carlo approach for integration of the rendering equation. In Boyanov, T., Dimova, S., Georgiev, K., and Nikolov, G., editors, *Numerical Methods and Applications*, pages 140–147, Berlin, Heidelberg. Springer Berlin Heidelberg.

[18] Donzis, D. A., Yeung, P. K., and Sreenivasan, K. R. (2008). Dissipation and enstrophy in isotropic turbulence: Resolution effects and scaling in direct numerical simulations. *Phys. Fluids*, 20(4):045108.

[19] Dopazo, C., Martin, J., Cifuentes, L., and Hierro, J. (2018). Strain, Rotation and Curvature of Non-material Propagating Iso-scalar Surfaces in Homogeneous Turbulence. *Flow Turbul. Combust.*, 101(1):1–32.

[20] Epanechnikov, V. (1969). Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158.

[21] Eswaran, V. and Pope, S. B. (1988). Direct numerical simulations of the turbulent mixing of a passive scalar. *Phys. Fluids*, 31:506–520.

[22] Freedman, D. and Diaconis, P. (1981). On the histogram as a density estimator:l2 theory. *Z. Wahrscheinlichkeit.*, 57(4):453–476.

[23] Gulitski, G., Kholmyansky, M., Kinzelbach, W., Lüthi, B., Tsinober, A., and Yorish, S. (2007a). Velocity and temperature derivatives in high-Reynolds-number turbulent flows in the atmospheric surface layer. Part 2. Accelerations and related matters. *J. Fluid Mech.*, 589:83–102.

[24] Gulitski, G., Kholmyansky, M., Kinzelbach, W., Lüthi, B., Tsinober, A., and Yorish, S. (2007b). Velocity and temperature derivatives in high-Reynolds-number turbulent flows in the atmospheric surface layer. Part 3. Temperature and joint statistics of temperature and velocity derivatives. *J. Fluid Mech.*, 589:103–123.

[25] Ishihara, T., Gotoh, T., and Kaneda, Y. (2009). Study of high-Reynolds number isotropic turbulence by direct numerical simulation. *Annu. Rev. Fluid Mech.*, 41:165–180.

[26] Ishihara, T., Kaneda, Y., Yokokawa, M., Itakura, K., and Uno, A. (2007). Small-scale statistics in high-resolution direct numerical simulation of turbulence: Reynolds number dependence of one-point velocity gradient statistics. *J. Fluid Mech.*, 592:335–366.

[27] Ishihara, T., Morishita, K., Yokokawa, M., Uno, A., and Kaneda, Y. (2016). Energy spectrum in high-resolution direct numerical simulations of turbulence. *Phys. Rev. Fluids*, 1:082403.

[28] Kahan, W. (1965). Pracniques: further remarks on reducing truncation errors. *Commun. ACM*, 8(1):40.

[29] Kim, S. H. and Bilger, R. W. (2007a). Iso-surface mass flow density and its implications for turbulent mixing and combustion. *Journal of Fluid Mechanics*.

[30] Kim, S. Y. and Bilger, R. W. (2007b). Iso-surface mass flow density and its implications for turbulent mixing and combustion. *J. Fluid Mech.*, 590:381–409.

[31] Kolla, H. and Chen, J. H. (2018). *Turbulent Combustion Simulations with High-Performance Computing*, pages 73–97. Springer Singapore, Singapore.

[32] Kolmogorov, A. N. (1941). Local structure of turbulence in an incompressible fluid at very high Reynolds numbers. *Dokl. Akad. Nauk SSSR*, 30:299–303.

[33] Lewiner, T., Lopes, H., Vieira, A. W., and Tavares, G. (2003). Efficient implementation of marching cubes' cases with topological guarantees. *J. Graph. Tools*, 8(2):1–15.

[34] Liu, Y. S., Yi, J., Zhang, H., Zheng, G. Q., and Paul, J. C. (2010). Surface area estimation of digitized 3D objects using quasi-Monte Carlo methods. *Pattern Recognition*.

[35] Muschinski, A. and de Bruyn Kops, S. M. (2015). Investigation of Hill's optical turbulence model by means of direct numerical simulation. *J. Opt. Soc. Am. A*, 32(12):2423–2430.

[36] Newman, T. S. and Yi, H. (2006). A survey of the marching cubes algorithm. *Comput. Graph.*, 30(5):854 – 879.

[37] Oboukhov, A. M. (1941a). Spectral energy distribution in a turbulent flow. *Dokl. Akad. Nauk. SSSR*, 32:22–24.

[38] Oboukhov, A. M. (1941b). Spectral energy distribution in a turbulent flow. *Izv. Akad. Nauk. SSSR, Ser. Geogr. i. Geofiz*, 5:453–466.

[39] Oboukhov, A. M. (1949). Structure of temperature field in a turbulent flow. *Izv. Akad. Nauk. SSSR, Geogr. i Geofiz*, 13:58–69.

[40] O'Neill, P. L. and Soria, J. (2005). The topology of homogeneous isotropic turbulence with passive scalar transport. In May, R. and Roberts, A. J., editors, *Proc. of 12th Computational Techniques and Applications Conference, CTAC-2004*, volume 46 of *ANZIAM J.*, pages C1170–C1187.

[41] Overholt, M. R. and Pope, S. B. (1998). A deterministic forcing scheme for direct numerical simulations of turbulence. *Comput. Fluids*, 27:11–28.

[42] Overholt, M. R. and Pope, S. B. (1999). Direct numerical simulation of a statistically stationary, turbulent reacting flow. *Combust. Theory Modelling*, 3(99):371–408.

[43] Papoulis (1965). *Probability, Random Variables, and Stochastic Processes*. McGraw Hill Inc.

[44] Patera, J. and Skala, V. (2004). A comparison of fundamental methods for iso surface extraction. *Machine Graphics and Vision*, 13(4):329–343.

[45] Payne, J. L., Hassan, B., and Af I, . S. (1988). Massively Parallel Computational Fluid Dynamics Calculations for Aerodynamics and Aerothermodynamics Applications. Technical report.

[46] Peters, N. (2000). *Turbulent combustion*. Cambridge university press.

[47] Pope, S. (1988). The evolution of surfaces in turbulence. *Int. J. Eng. Sci.*, 26(5):445 – 469.

[48] Pope, S. B. (1991). Computations of turbulent combustion: Progress and challenges. *Symposium (International) on Combustion*.

[49] Pope, S. B. (2000). *Turbulent Flows*. Cambridge University Press, Cambridge.

[50] Pope, S. B., Yeung, P. K., and Girimaji, S. S. (1989). The curvature of material surfaces in isotropic turbulence. *Phys. Fluids A-Fluid*, 1(12):2010–2018.

[51] Portwood, G. D., de Bruyn Kops, S. M., and Caulfield, C. P. (submitted). Asymptotic dynamics of high dynamic range stratified turbulence. *Phys. Rev. Lett.*

[52] Portwood, G. D., de Bruyn Kops, S. M., Taylor, J. R., Salehipour, H., and Caulfield, C. P. (2016). Robust identification of dynamically distinct regions in stratified turbulence. *J. Fluid Mech.*, 807:R2 (14 pages).

[53] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition.

[54] Raase, S. and Nordström, T. (2015). On the use of a many-core processor for computational fluid dynamics simulations. In *Procedia Comput. Sci.*

[55] Rao, K. J. and de Bruyn Kops, S. M. (2011). A mathematical framework for forcing turbulence applied to horizontally homogeneous stratified flow. *Phys. Fluids*, 23:065110.

[56] Richardson, L. F. (1922). *Weather Prediction by Numerical Process*. Cambridge University Press.

[57] Ricou, F. P. and Spalding, D. B. (1961). Measurements of entrainment by axisymmetrical turbulent jets. *J. Fluid Mech.*, 11(1):21–32.

[58] Scheidegger, C. E., Schreiner, J. M., Duffy, B., Carr, H., and Silva, C. T. (2008). Revisiting histograms and isosurface statistics. *IEEE T. Vis. Comput. Gr.*, 14(6):1659–1666.

[59] Schmidt, D. P. and Bedford, F. (2018). An analysis of the convergence of stochastic Lagrangian/Eulerian spray simulations. *International Journal of Multiphase Flow*, 102:95–101.

[60] Schroeder, W., Martin, K., Lorensen, B., Avila, L. S., Avila, R., and Law, C. C. (2006). The Visualization Toolkit An Object-Oriented Approach To 3D Graphics Fourth Edition. Technical report.

[61] Schumacher, J. and Sreenivasan, K. R. (2005). Statistics and geometry of passive scalars in turbulence. *Phys. Fluids*.

[62] Schumacher, J., Sreenivasan, K. R., and Yakhot, V. (2007). Asymptotic exponents from low-Reynolds-number flows. *New J. Phys.*, 9:89.

[63] Schumacher, J., Sreenivasan, K. R., and Yeung, P. K. (2005). Very fine structures in scalar mixing. *J. Fluid Mech.*, 531:113–122.

[64] Sobol', I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802.

[65] Sreenivasan, K. R. (1991). On local isotropy of passive scalars in turbulent shear flows. *Proc. R. Soc. Lond. A*, 434(1890):165–182.

[66] Sreenivasan, K. R. (2018). Turbulent mixing: A perspective. *Proceedings of the National Academy of Sciences*, 2018:201800463.

[67] Storti, D. (2019). Likelihood of level sets. *Personal Communications*.

[68] Swaminathan, N. and Bilger, R. W. (1997). Direct numerical simulation of turbulent nonpremixed hydrocarbon flames using a two-step reduced mechanism. *Combustion Science and Technology*, 127(1-6):167–196.

[69] Swaminathan, N. and Bray, K. (2011). *Turbulent Premixed Flames*. Cambridge University Press.

[70] Taveira, R. R. and da Silva, C. B. (2014). Characteristics of the viscous superlayer in shear free turbulence and in planar turbulent jets. *Phys. Fluids*, 26(2):021702.

[71] Vadhan, S. P. et al. (2012). Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336.

[72] Watanabe, T. and Gotoh, T. (2007). Inertial-range intermittency and accuracy of direct numerical simulation for turbulence and passive scalar turbulence. *J. Fluid Mech.*, 590:117–146.

[73] Watanabe, T., Riley, J. J., de Bruyn Kops, S. M., Diamessis, P. J., and Zhou, Q. (2016). Turbulent/non-turbulent interfaces in wakes in stably stratified fluids. *J. Fluid Mech.*, 797:R1.

[74] Yakhot, V. and Sreenivasan, K. R. (2005). Anomalous scaling of structure functions and dynamic constraints on turbulence simulations. *J. Stat. Phys.*, 121:823–841.

[75] Yeung, P., Sreenivasan, K., and Pope, S. (2018). Effects of finite spatial and temporal resolution in direct numerical simulations of incompressible isotropic turbulence. *Phys. Rev. Fluids*, 3(6):064603.

[76] Yeung, P. K., Donzis, D. A., and Sreenivasan, K. R. (2005). High-Reynolds-number simulation of turbulent mixing. *Phys. Fluids*, 17:081703.

[77] Yeung, P. K. and Sawford, B. L. (2002). Random-sweeping hypothesis for passive scalars in isotropic turbulence. *J. Fluid Mech.*, 459:129–138.

[78] Yurtoglu, M., Carton, M., and Storti, D. (2018). Treat all integrals as volume integrals: A unified, parallel, grid-based method for evaluation of volume, surface, and path integrals on implicitly defined domains. *J. Inf. Sci. Eng.*, 18(2):021013.

[79] Zheng, T., You, J., and Yang, Y. (2017). Principal curvatures and area ratio of propagating surfaces in isotropic turbulence. *Phys. Rev. Fluids*, 2:103201.